HEXAWARE

ASSIGNMENT 3

Tasks 1: Database Design:

1. Create the database named "HMBank".

2. Define the schema for the Customers, Accounts, and Transactions tables based on the provided schema

4. Create an ERD

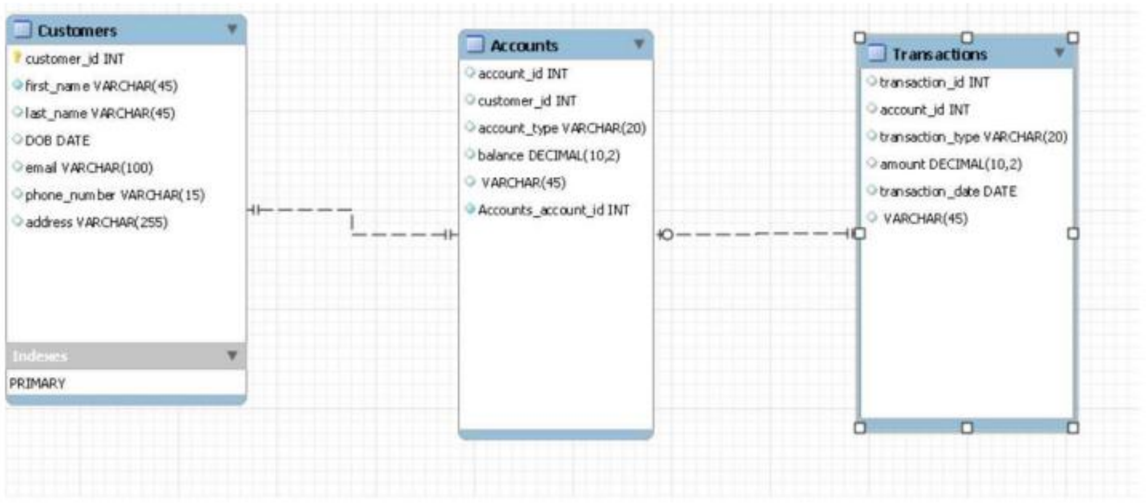
5. Create appropriate Primary Key and Foreign Key constraints for referential integrity.

6. Write SQL scripts to create the mentioned tables with appropriate data types, constraints, and relationships.

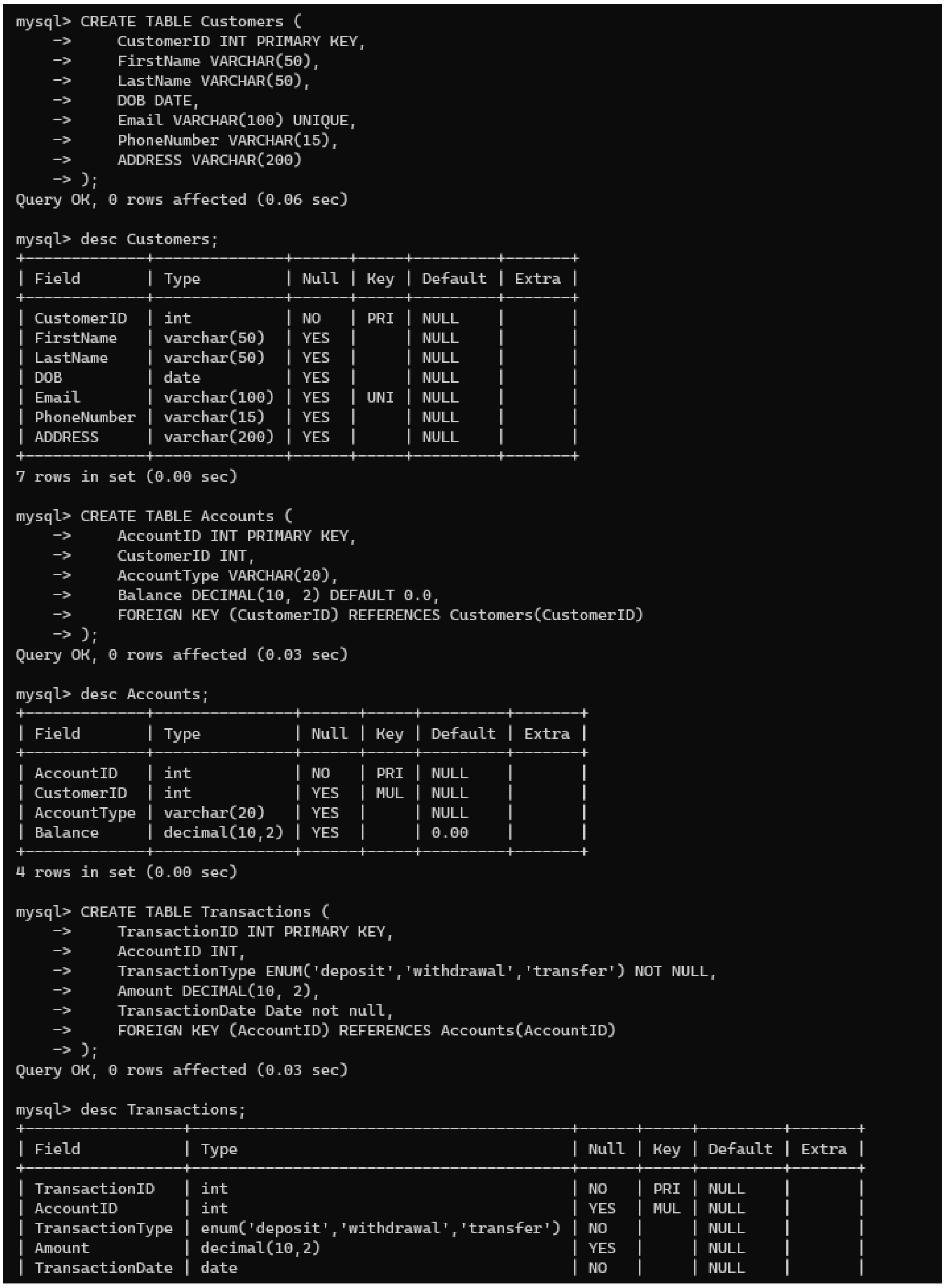
• Customers

• Accounts

• Transactions



ERD Diagram

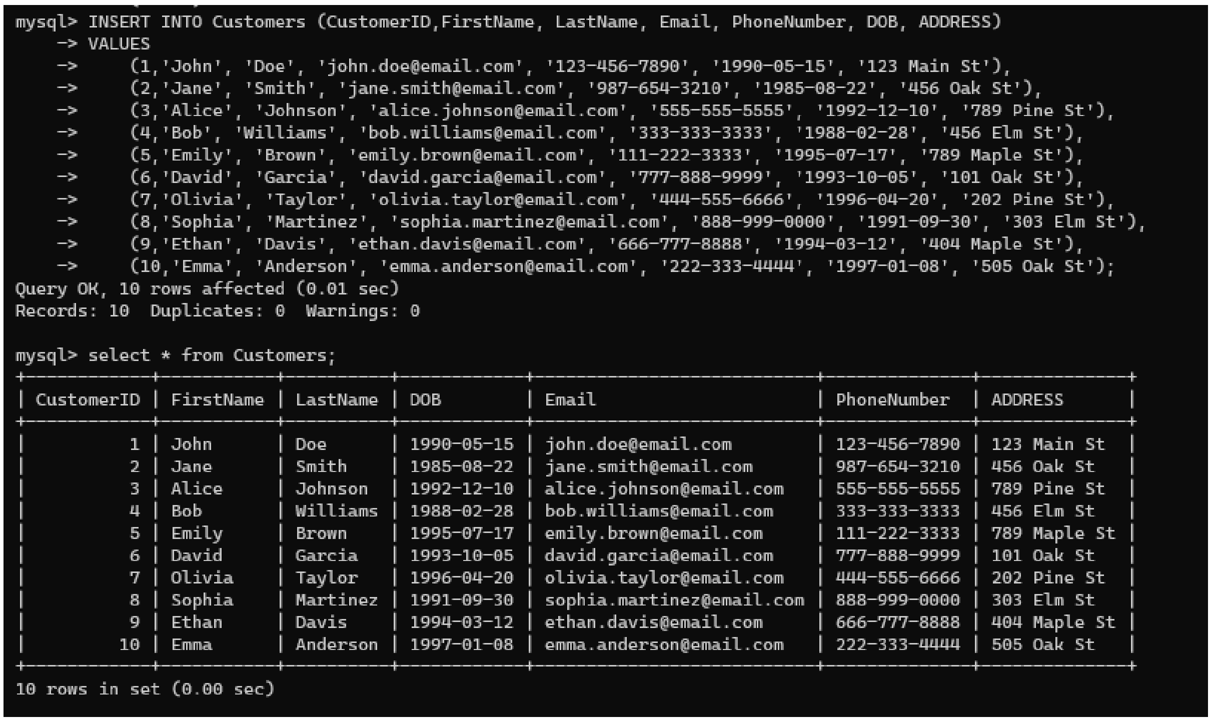


```
mysql> CREATE TABLE Customers (
    ->     CustomerID INT PRIMARY KEY,
    ->     FirstName VARCHAR(50),
    ->     LastName VARCHAR(50),
    ->     DOB DATE,
    ->     Email VARCHAR(100) UNIQUE,
    ->     PhoneNumber VARCHAR(15),
    ->     ADDRESS VARCHAR(200)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> desc Customers;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| CustomerID  | int          | NO   | PRI | NULL    |       |
| FirstName   | varchar(50)  | YES  |     | NULL    |       |
| LastName    | varchar(50)  | YES  |     | NULL    |       |
| DOB         | date         | YES  |     | NULL    |       |
| Email       | varchar(100) | YES  | UNI | NULL    |       |
| PhoneNumber | varchar(15)  | YES  |     | NULL    |       |
| ADDRESS     | varchar(200) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql> CREATE TABLE Accounts (
    ->     AccountID INT PRIMARY KEY,
    ->     CustomerID INT,
    ->     AccountType VARCHAR(20),
    ->     Balance DECIMAL(10, 2) DEFAULT 0.0,
    ->     FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> desc Accounts;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| AccountID   | int           | NO   | PRI | NULL    |       |
| CustomerID  | int           | YES  | MUL | NULL    |       |
| AccountType | varchar(20)   | YES  |     | NULL    |       |
| Balance     | decimal(10,2) | YES  |     | 0.00    |       |
+-------------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> CREATE TABLE Transactions (
    ->     TransactionID INT PRIMARY KEY,
    ->     AccountID INT,
    ->     TransactionType ENUM('deposit','withdrawal','transfer') NOT NULL,
    ->     Amount DECIMAL(10, 2),
    ->     TransactionDate Date not null,
    ->     FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> desc Transactions;
+-----------------+------------------------------------------+------+-----+---------+-------+
| Field           | Type                                     | Null | Key | Default | Extra |
+-----------------+------------------------------------------+------+-----+---------+-------+
| TransactionID   | int                                      | NO   | PRI | NULL    |       |
| AccountID       | int                                      | YES  | MUL | NULL    |       |
| TransactionType | enum('deposit','withdrawal','transfer')  | NO   |     | NULL    |       |
| Amount          | decimal(10,2)                            | YES  |     | NULL    |       |
| TransactionDate | date                                     | NO   |     | NULL    |       |
```

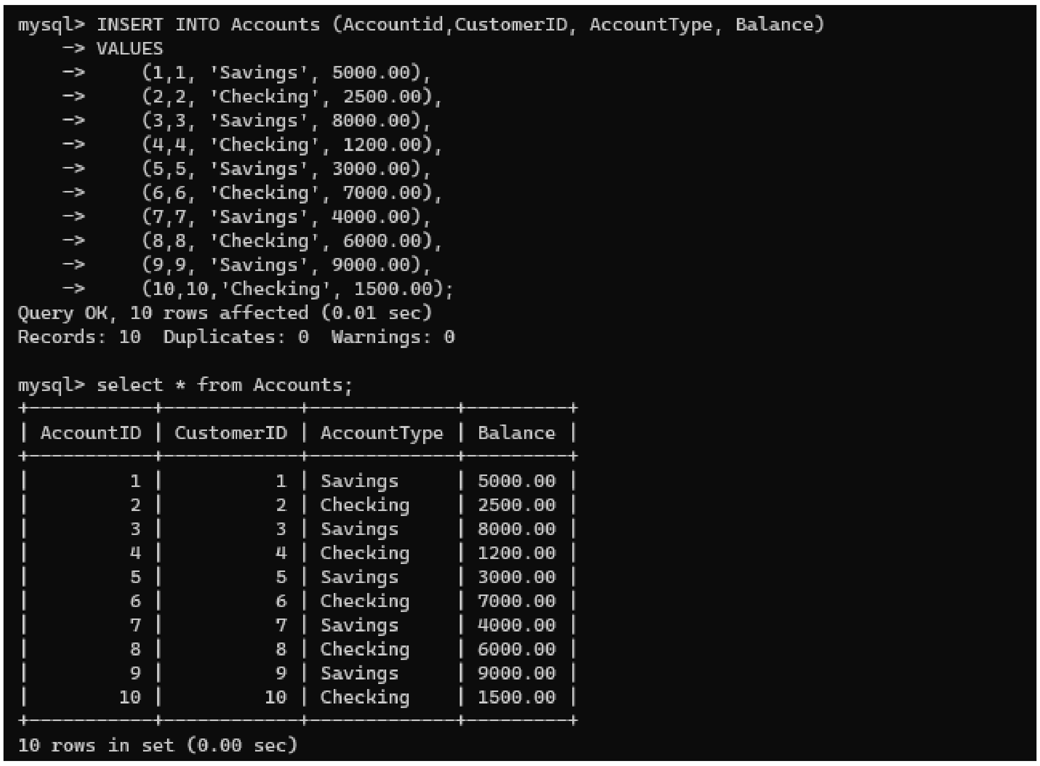
Tasks 2: Select, Where, Between, AND, LIKE:

1. Insert at least 10 sample records into each of the following tables.

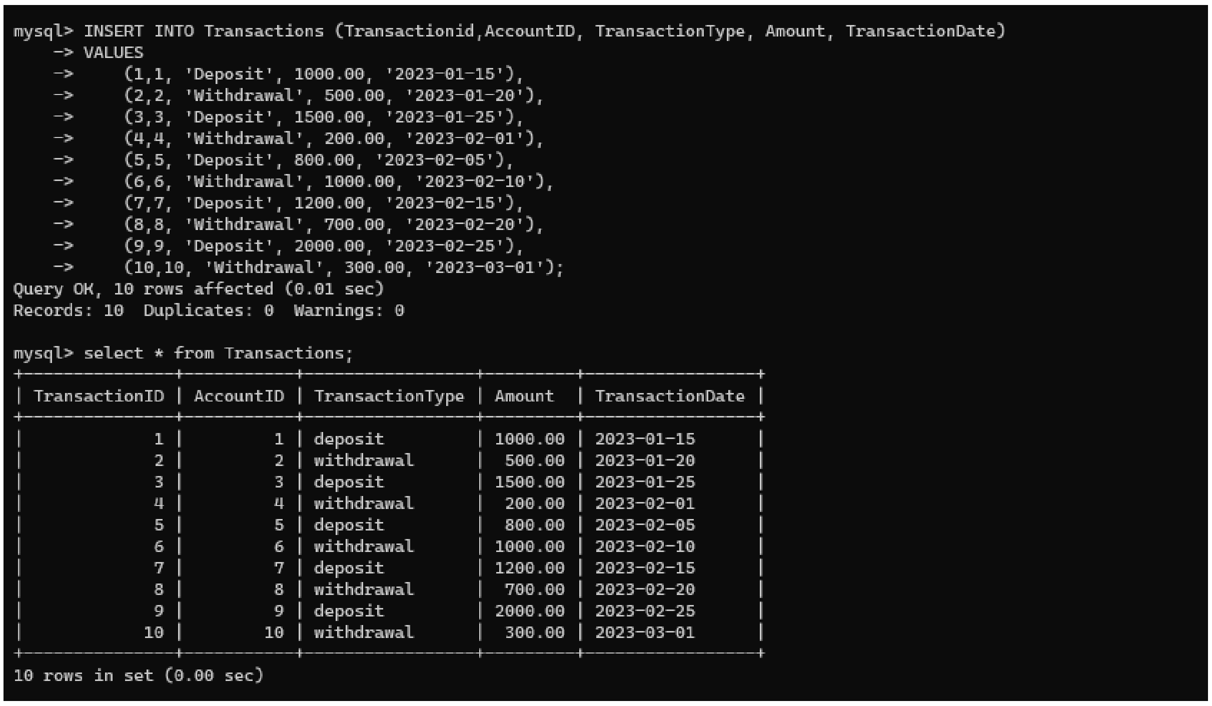
- Customers
- Accounts
- Transactions



```
mysql> INSERT INTO Customers (CustomerID,FirstName, LastName, Email, PhoneNumber, DOB, ADDRESS)
    -> VALUES
    ->     (1,'John', 'Doe', 'john.doe@email.com', '123-456-7890', '1990-05-15', '123 Main St'),
    ->     (2,'Jane', 'Smith', 'jane.smith@email.com', '987-654-3210', '1985-08-22', '456 Oak St'),
    ->     (3,'Alice', 'Johnson', 'alice.johnson@email.com', '555-555-5555', '1992-12-10', '789 Pine St'),
    ->     (4,'Bob', 'Williams', 'bob.williams@email.com', '333-333-3333', '1988-02-28', '456 Elm St'),
    ->     (5,'Emily', 'Brown', 'emily.brown@email.com', '111-222-3333', '1995-07-17', '789 Maple St'),
    ->     (6,'David', 'Garcia', 'david.garcia@email.com', '777-888-9999', '1993-10-05', '101 Oak St'),
    ->     (7,'Olivia', 'Taylor', 'olivia.taylor@email.com', '444-555-6666', '1996-04-20', '202 Pine St'),
    ->     (8,'Sophia', 'Martinez', 'sophia.martinez@email.com', '888-999-0000', '1991-09-30', '303 Elm St'),
    ->     (9,'Ethan', 'Davis', 'ethan.davis@email.com', '666-777-8888', '1994-03-12', '404 Maple St'),
    ->     (10,'Emma', 'Anderson', 'emma.anderson@email.com', '222-333-4444', '1997-01-08', '505 Oak St');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Customers;
+------------+-----------+----------+------------+---------------------------+--------------+--------------+
| CustomerID | FirstName | LastName | DOB        | Email                     | PhoneNumber  | ADDRESS      |
+------------+-----------+----------+------------+---------------------------+--------------+--------------+
|          1 | John      | Doe      | 1990-05-15 | john.doe@email.com        | 123-456-7890 | 123 Main St  |
|          2 | Jane      | Smith    | 1985-08-22 | jane.smith@email.com      | 987-654-3210 | 456 Oak St   |
|          3 | Alice     | Johnson  | 1992-12-10 | alice.johnson@email.com   | 555-555-5555 | 789 Pine St  |
|          4 | Bob       | Williams | 1988-02-28 | bob.williams@email.com    | 333-333-3333 | 456 Elm St   |
|          5 | Emily     | Brown    | 1995-07-17 | emily.brown@email.com     | 111-222-3333 | 789 Maple St |
|          6 | David     | Garcia   | 1993-10-05 | david.garcia@email.com    | 777-888-9999 | 101 Oak St   |
|          7 | Olivia    | Taylor   | 1996-04-20 | olivia.taylor@email.com   | 444-555-6666 | 202 Pine St  |
|          8 | Sophia    | Martinez | 1991-09-30 | sophia.martinez@email.com | 888-999-0000 | 303 Elm St   |
|          9 | Ethan     | Davis    | 1994-03-12 | ethan.davis@email.com     | 666-777-8888 | 404 Maple St |
|         10 | Emma      | Anderson | 1997-01-08 | emma.anderson@email.com   | 222-333-4444 | 505 Oak St   |
+------------+-----------+----------+------------+---------------------------+--------------+--------------+
10 rows in set (0.00 sec)
```

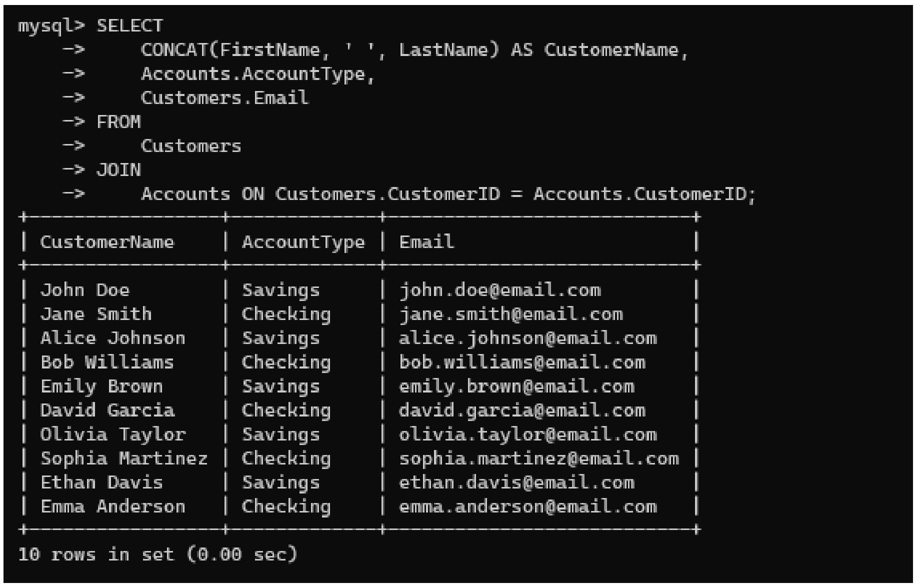


```
mysql> INSERT INTO Accounts (Accountid,CustomerID, AccountType, Balance)
    -> VALUES
    ->     (1,1, 'Savings', 5000.00),
    ->     (2,2, 'Checking', 2500.00),
    ->     (3,3, 'Savings', 8000.00),
    ->     (4,4, 'Checking', 1200.00),
    ->     (5,5, 'Savings', 3000.00),
    ->     (6,6, 'Checking', 7000.00),
    ->     (7,7, 'Savings', 4000.00),
    ->     (8,8, 'Checking', 6000.00),
    ->     (9,9, 'Savings', 9000.00),
    ->     (10,10,'Checking', 1500.00);
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Accounts;
+-----------+------------+-------------+---------+
| AccountID | CustomerID | AccountType | Balance |
+-----------+------------+-------------+---------+
|         1 |          1 | Savings     | 5000.00 |
|         2 |          2 | Checking    | 2500.00 |
|         3 |          3 | Savings     | 8000.00 |
|         4 |          4 | Checking    | 1200.00 |
|         5 |          5 | Savings     | 3000.00 |
|         6 |          6 | Checking    | 7000.00 |
|         7 |          7 | Savings     | 4000.00 |
|         8 |          8 | Checking    | 6000.00 |
|         9 |          9 | Savings     | 9000.00 |
|        10 |         10 | Checking    | 1500.00 |
+-----------+------------+-------------+---------+
10 rows in set (0.00 sec)
```



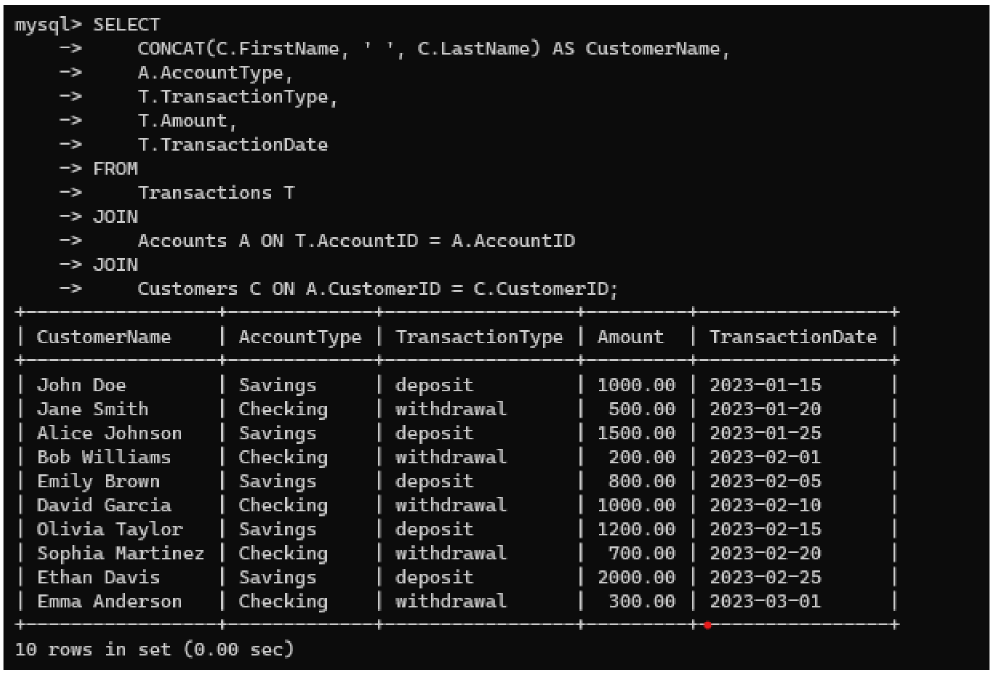
```
mysql> INSERT INTO Transactions (Transactionid,AccountID, TransactionType, Amount, TransactionDate)
    -> VALUES
    ->     (1,1, 'Deposit', 1000.00, '2023-01-15'),
    ->     (2,2, 'Withdrawal', 500.00, '2023-01-20'),
    ->     (3,3, 'Deposit', 1500.00, '2023-01-25'),
    ->     (4,4, 'Withdrawal', 200.00, '2023-02-01'),
    ->     (5,5, 'Deposit', 800.00, '2023-02-05'),
    ->     (6,6, 'Withdrawal', 1000.00, '2023-02-10'),
    ->     (7,7, 'Deposit', 1200.00, '2023-02-15'),
    ->     (8,8, 'Withdrawal', 700.00, '2023-02-20'),
    ->     (9,9, 'Deposit', 2000.00, '2023-02-25'),
    ->     (10,10, 'Withdrawal', 300.00, '2023-03-01');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> select * from Transactions;
+---------------+-----------+-----------------+---------+-----------------+
| TransactionID | AccountID | TransactionType | Amount  | TransactionDate |
+---------------+-----------+-----------------+---------+-----------------+
|             1 |         1 | deposit         | 1000.00 | 2023-01-15      |
|             2 |         2 | withdrawal      |  500.00 | 2023-01-20      |
|             3 |         3 | deposit         | 1500.00 | 2023-01-25      |
|             4 |         4 | withdrawal      |  200.00 | 2023-02-01      |
|             5 |         5 | deposit         |  800.00 | 2023-02-05      |
|             6 |         6 | withdrawal      | 1000.00 | 2023-02-10      |
|             7 |         7 | deposit         | 1200.00 | 2023-02-15      |
|             8 |         8 | withdrawal      |  700.00 | 2023-02-20      |
|             9 |         9 | deposit         | 2000.00 | 2023-02-25      |
|            10 |        10 | withdrawal      |  300.00 | 2023-03-01      |
+---------------+-----------+-----------------+---------+-----------------+
10 rows in set (0.00 sec)
```

2. Write a SQL query to retrieve the name, account type and email of all customers.



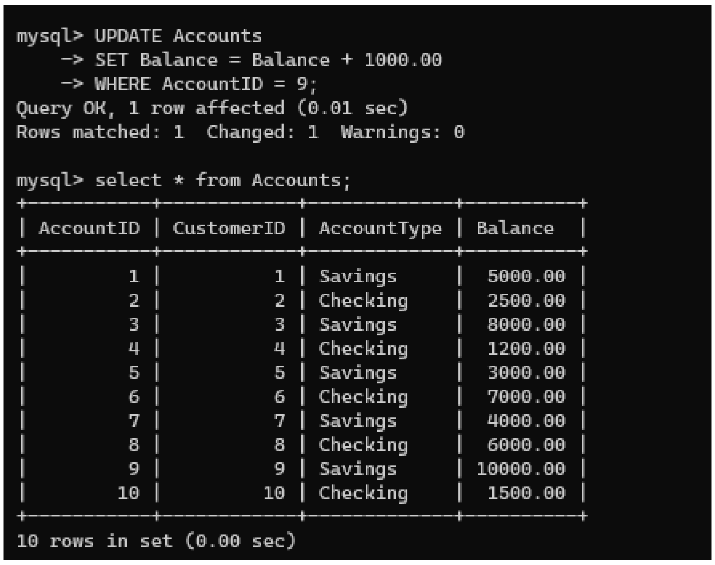
```
mysql> SELECT
    ->     CONCAT(FirstName, ' ', LastName) AS CustomerName,
    ->     Accounts.AccountType,
    ->     Customers.Email
    -> FROM
    ->     Customers
    -> JOIN
    ->     Accounts ON Customers.CustomerID = Accounts.CustomerID;
+-----------------+-------------+---------------------------+
| CustomerName    | AccountType | Email                     |
+-----------------+-------------+---------------------------+
| John Doe        | Savings     | john.doe@email.com        |
| Jane Smith      | Checking    | jane.smith@email.com      |
| Alice Johnson   | Savings     | alice.johnson@email.com   |
| Bob Williams    | Checking    | bob.williams@email.com    |
| Emily Brown     | Savings     | emily.brown@email.com     |
| David Garcia    | Checking    | david.garcia@email.com    |
| Olivia Taylor   | Savings     | olivia.taylor@email.com   |
| Sophia Martinez | Checking    | sophia.martinez@email.com |
| Ethan Davis     | Savings     | ethan.davis@email.com     |
| Emma Anderson   | Checking    | emma.anderson@email.com   |
+-----------------+-------------+---------------------------+
10 rows in set (0.00 sec)
```

2. Write a SQL query to retrieve the name, account type and email of all customers.



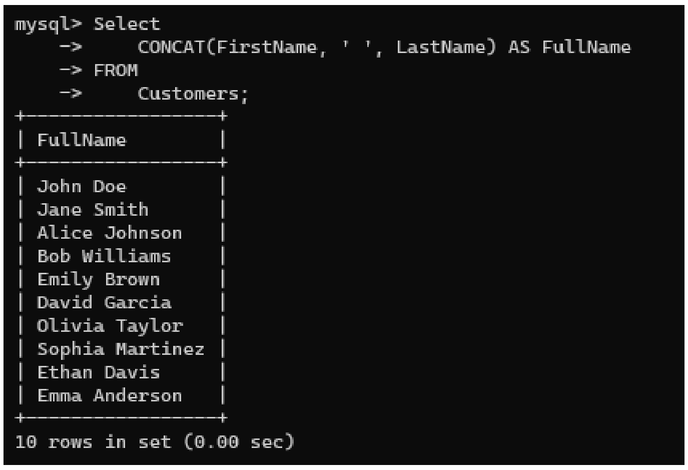
```
mysql> SELECT
    ->     CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName,
    ->     A.AccountType,
    ->     T.TransactionType,
    ->     T.Amount,
    ->     T.TransactionDate
    -> FROM
    ->     Transactions T
    -> JOIN
    ->     Accounts A ON T.AccountID = A.AccountID
    -> JOIN
    ->     Customers C ON A.CustomerID = C.CustomerID;
+-----------------+-------------+-----------------+---------+-----------------+
| CustomerName    | AccountType | TransactionType | Amount  | TransactionDate |
+-----------------+-------------+-----------------+---------+-----------------+
| John Doe        | Savings     | deposit         | 1000.00 | 2023-01-15      |
| Jane Smith      | Checking    | withdrawal      |  500.00 | 2023-01-20      |
| Alice Johnson   | Savings     | deposit         | 1500.00 | 2023-01-25      |
| Bob Williams    | Checking    | withdrawal      |  200.00 | 2023-02-01      |
| Emily Brown     | Savings     | deposit         |  800.00 | 2023-02-05      |
| David Garcia    | Checking    | withdrawal      | 1000.00 | 2023-02-10      |
| Olivia Taylor   | Savings     | deposit         | 1200.00 | 2023-02-15      |
| Sophia Martinez | Checking    | withdrawal      |  700.00 | 2023-02-20      |
| Ethan Davis     | Savings     | deposit         | 2000.00 | 2023-02-25      |
| Emma Anderson   | Checking    | withdrawal      |  300.00 | 2023-03-01      |
+-----------------+-------------+-----------------+---------+-----------------+
10 rows in set (0.00 sec)
```

4. Write a SQL query to increase the balance of a specific account by a certain amount



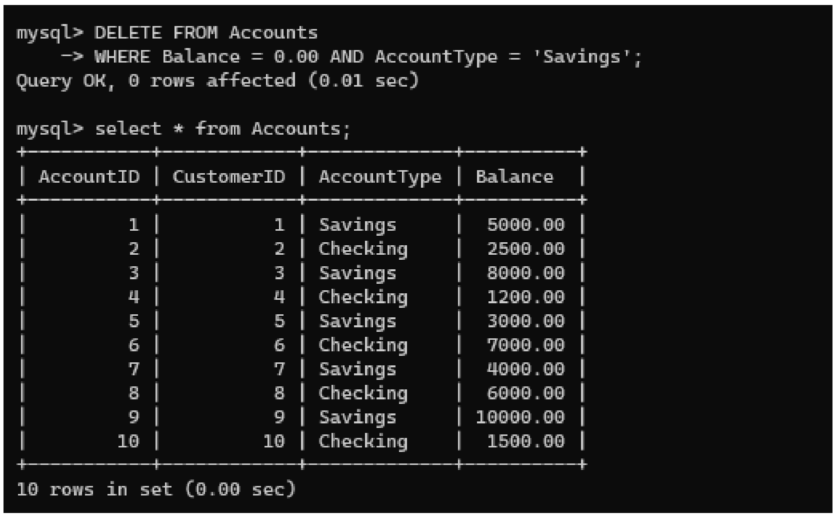
```
mysql> UPDATE Accounts
    -> SET Balance = Balance + 1000.00
    -> WHERE AccountID = 9;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from Accounts;
+-----------+------------+-------------+----------+
| AccountID | CustomerID | AccountType | Balance  |
+-----------+------------+-------------+----------+
|         1 |          1 | Savings     |  5000.00 |
|         2 |          2 | Checking    |  2500.00 |
|         3 |          3 | Savings     |  8000.00 |
|         4 |          4 | Checking    |  1200.00 |
|         5 |          5 | Savings     |  3000.00 |
|         6 |          6 | Checking    |  7000.00 |
|         7 |          7 | Savings     |  4000.00 |
|         8 |          8 | Checking    |  6000.00 |
|         9 |          9 | Savings     | 10000.00 |
|        10 |         10 | Checking    |  1500.00 |
+-----------+------------+-------------+----------+
10 rows in set (0.00 sec)
```

5. Write a SQL query to Combine first and last names of customers as a full name.



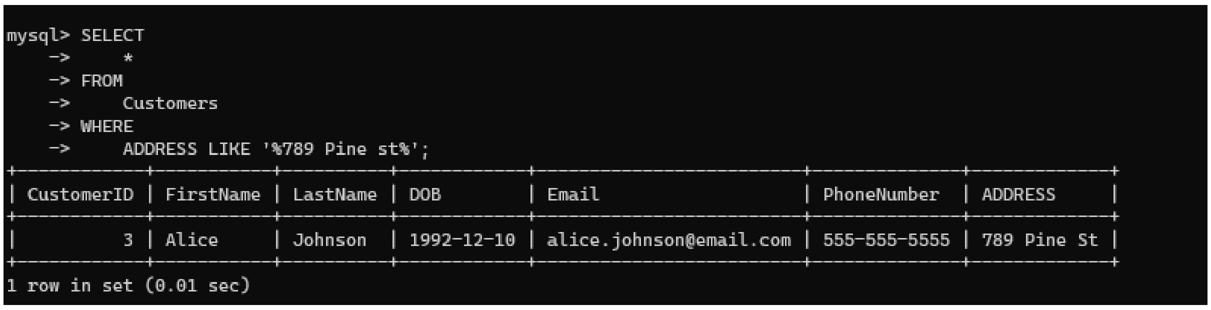
```
mysql> Select
    ->     CONCAT(FirstName, ' ', LastName) AS FullName
    -> FROM
    ->     Customers;
+-----------------+
| FullName        |
+-----------------+
| John Doe        |
| Jane Smith      |
| Alice Johnson   |
| Bob Williams    |
| Emily Brown     |
| David Garcia    |
| Olivia Taylor   |
| Sophia Martinez |
| Ethan Davis     |
| Emma Anderson   |
+-----------------+
10 rows in set (0.00 sec)
```

6. Write a SQL query to remove accounts with a balance of zero where the account type is savings.



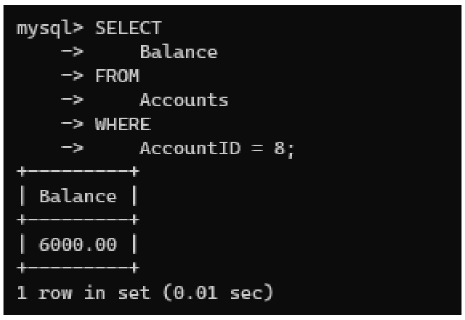
```
mysql> DELETE FROM Accounts
    -> WHERE Balance = 0.00 AND AccountType = 'Savings';
Query OK, 0 rows affected (0.01 sec)

mysql> select * from Accounts;
+-----------+------------+-------------+----------+
| AccountID | CustomerID | AccountType | Balance  |
+-----------+------------+-------------+----------+
|         1 |          1 | Savings     |  5000.00 |
|         2 |          2 | Checking    |  2500.00 |
|         3 |          3 | Savings     |  8000.00 |
|         4 |          4 | Checking    |  1200.00 |
|         5 |          5 | Savings     |  3000.00 |
|         6 |          6 | Checking    |  7000.00 |
|         7 |          7 | Savings     |  4000.00 |
|         8 |          8 | Checking    |  6000.00 |
|         9 |          9 | Savings     | 10000.00 |
|        10 |         10 | Checking    |  1500.00 |
+-----------+------------+-------------+----------+
10 rows in set (0.00 sec)
```

7. Write a SQL query to Find customers living in a specific city.



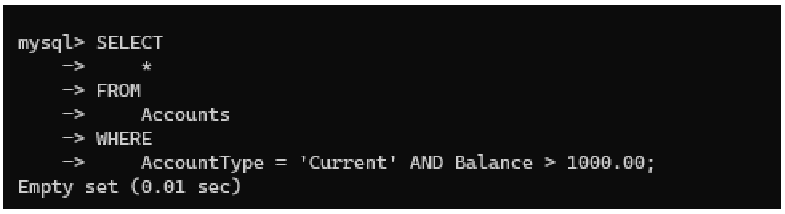
```
mysql> SELECT
    ->     *
    -> FROM
    ->     Customers
    -> WHERE
    ->     ADDRESS LIKE '%789 Pine st%';
+------------+-----------+----------+------------+-------------------------+--------------+-------------+
| CustomerID | FirstName | LastName | DOB        | Email                   | PhoneNumber  | ADDRESS     |
+------------+-----------+----------+------------+-------------------------+--------------+-------------+
|          3 | Alice     | Johnson  | 1992-12-10 | alice.johnson@email.com | 555-555-5555 | 789 Pine St |
+------------+-----------+----------+------------+-------------------------+--------------+-------------+
1 row in set (0.01 sec)
```

8. Write a SQL query to Get the account balance for a specific account.



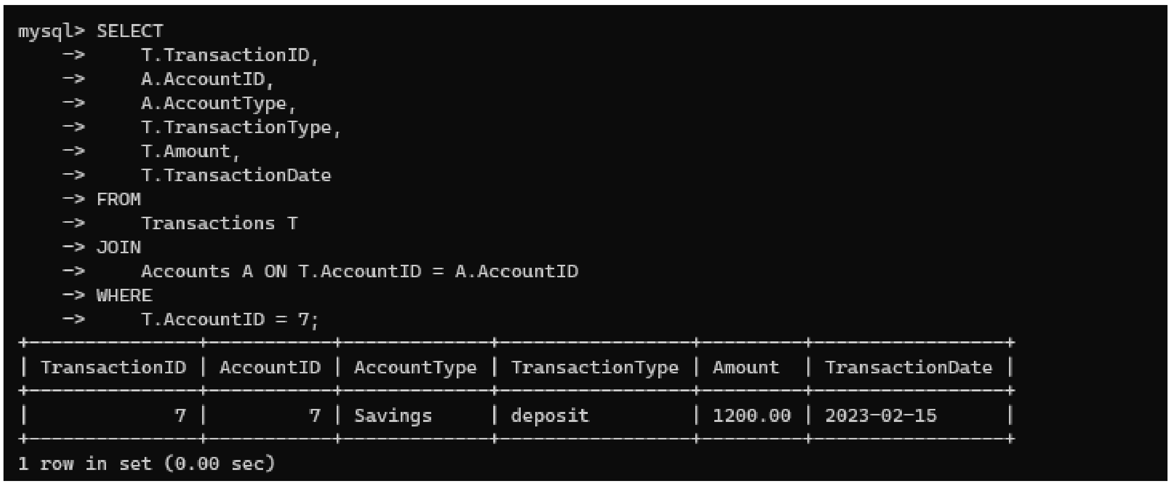
```
mysql> SELECT
    ->     Balance
    -> FROM
    ->     Accounts
    -> WHERE
    ->     AccountID = 8;
+---------+
| Balance |
+---------+
| 6000.00 |
+---------+
1 row in set (0.01 sec)
```

9. Write a SQL query to List all current accounts with a balance greater than $1,000.



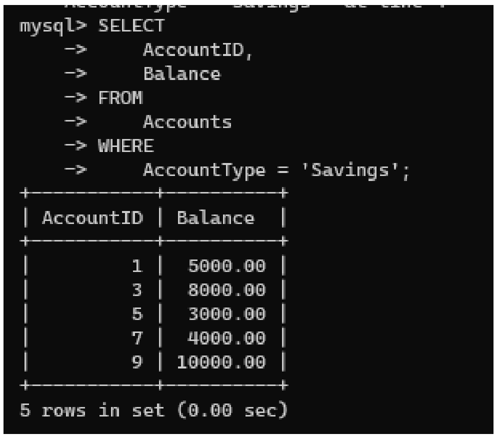
```
mysql> SELECT
    ->     *
    -> FROM
    ->     Accounts
    -> WHERE
    ->     AccountType = 'Current' AND Balance > 1000.00;
Empty set (0.01 sec)
```

10. Write a SQL query to Retrieve all transactions for a specific account.



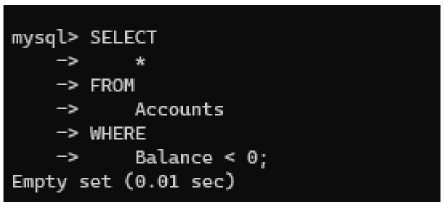
```
mysql> SELECT
    ->     T.TransactionID,
    ->     A.AccountID,
    ->     A.AccountType,
    ->     T.TransactionType,
    ->     T.Amount,
    ->     T.TransactionDate
    -> FROM
    ->     Transactions T
    -> JOIN
    ->     Accounts A ON T.AccountID = A.AccountID
    -> WHERE
    ->     T.AccountID = 7;
+---------------+-----------+-------------+-----------------+---------+-----------------+
| TransactionID | AccountID | AccountType | TransactionType | Amount  | TransactionDate |
+---------------+-----------+-------------+-----------------+---------+-----------------+
|             7 |         7 | Savings     | deposit         | 1200.00 | 2023-02-15      |
+---------------+-----------+-------------+-----------------+---------+-----------------+
1 row in set (0.00 sec)
```

11. Write a SQL query to Calculate the interest accrued on savings accounts based on a given interest rate.



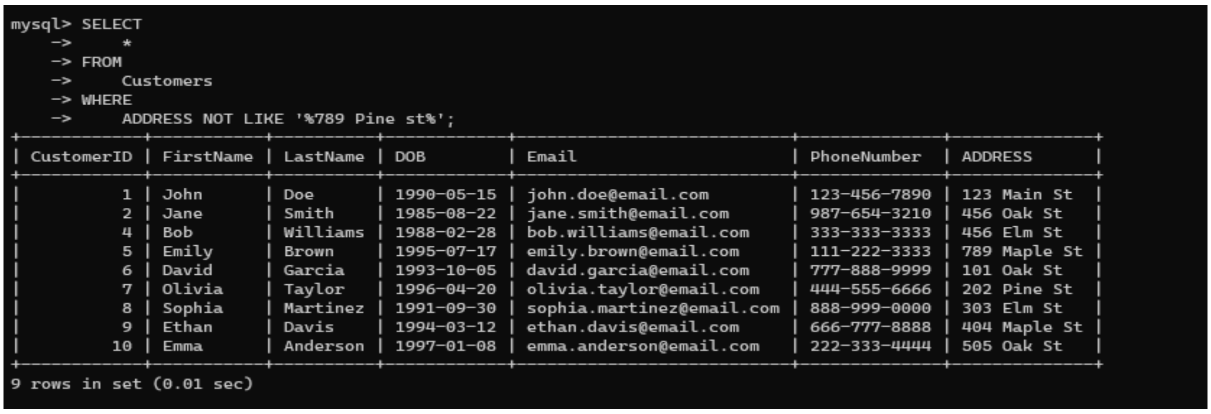
```
mysql> SELECT
    ->     AccountID,
    ->     Balance
    -> FROM
    ->     Accounts
    -> WHERE
    ->     AccountType = 'Savings';
+-----------+----------+
| AccountID | Balance  |
+-----------+----------+
|         1 |  5000.00 |
|         3 |  8000.00 |
|         5 |  3000.00 |
|         7 |  4000.00 |
|         9 | 10000.00 |
+-----------+----------+
5 rows in set (0.00 sec)
```

12. Write a SQL query to Identify accounts where the balance is less than a specified overdraft limit.



```
mysql> SELECT
    ->     *
    -> FROM
    ->     Accounts
    -> WHERE
    ->     Balance < 0;
Empty set (0.01 sec)
```

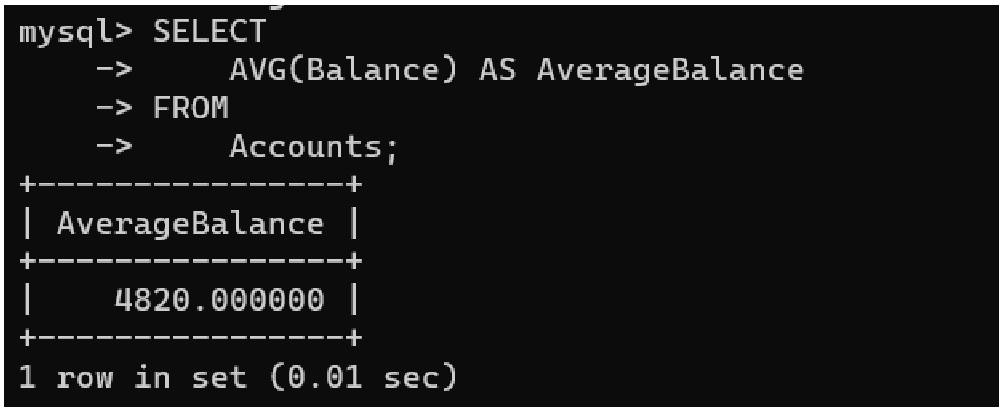
13. Write a SQL query to Find customers not living in a specific city.



```
mysql> SELECT
    ->     *
    -> FROM
    ->     Customers
    -> WHERE
    ->     ADDRESS NOT LIKE '%789 Pine st%';
+------------+-----------+-----------+------------+----------------------------+--------------+--------------+
| CustomerID | FirstName | LastName  | DOB        | Email                      | PhoneNumber  | ADDRESS      |
+------------+-----------+-----------+------------+----------------------------+--------------+--------------+
|          1 | John      | Doe       | 1990-05-15 | john.doe@email.com         | 123-456-7890 | 123 Main St  |
|          2 | Jane      | Smith     | 1985-08-22 | jane.smith@email.com       | 987-654-3210 | 456 Oak St   |
|          4 | Bob       | Williams  | 1988-02-28 | bob.williams@email.com     | 333-333-3333 | 456 Elm St   |
|          5 | Emily     | Brown     | 1995-07-17 | emily.brown@email.com      | 111-222-3333 | 789 Maple St |
|          6 | David     | Garcia    | 1993-10-05 | david.garcia@email.com     | 777-888-9999 | 101 Oak St   |
|          7 | Olivia    | Taylor    | 1996-04-20 | olivia.taylor@email.com    | 444-555-6666 | 202 Pine St  |
|          8 | Sophia    | Martinez  | 1991-09-30 | sophia.martinez@email.com  | 888-999-0000 | 303 Elm St   |
|          9 | Ethan     | Davis     | 1994-03-12 | ethan.davis@email.com      | 666-777-8888 | 404 Maple St |
|         10 | Emma      | Anderson  | 1997-01-08 | emma.anderson@email.com    | 222-333-4444 | 505 Oak St   |
+------------+-----------+-----------+------------+----------------------------+--------------+--------------+
9 rows in set (0.01 sec)
```

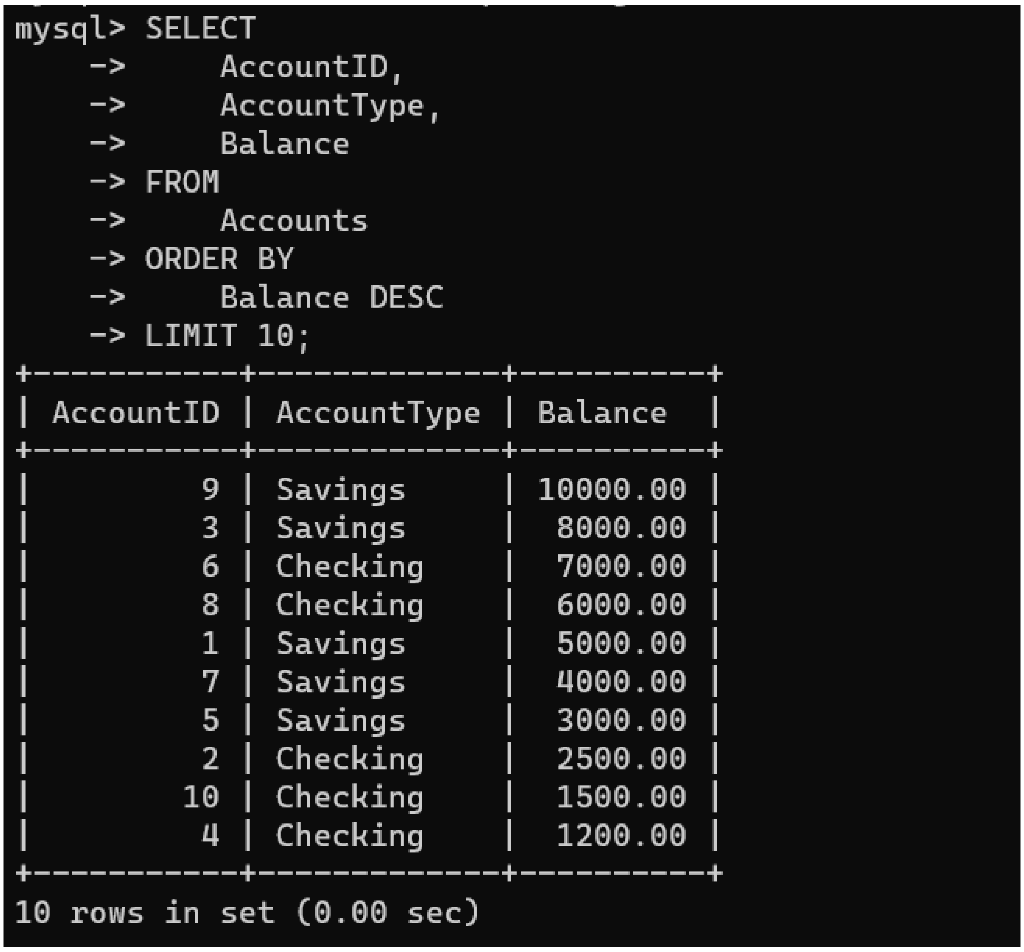
Tasks 3: Aggregate functions, Having, Order By, GroupBy and Joins:

1. Write a SQL query to Find the average account balance for all customers.



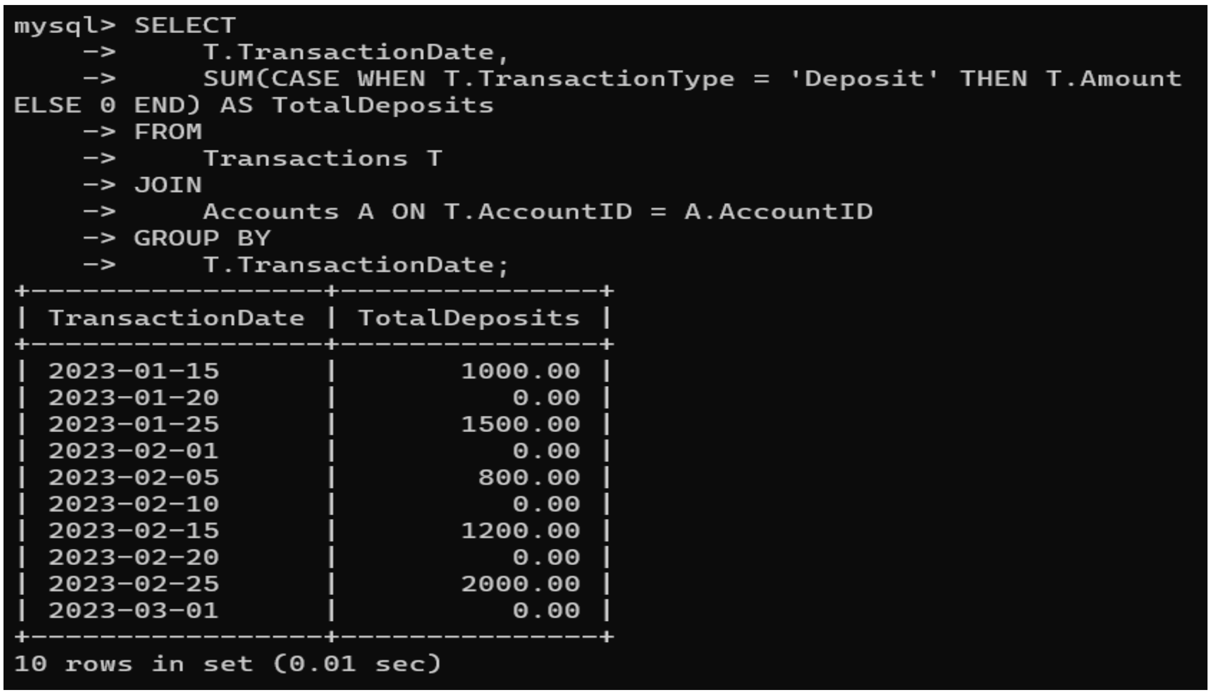
```
mysql> SELECT
    ->     AVG(Balance) AS AverageBalance
    -> FROM
    ->     Accounts;
+----------------+
| AverageBalance |
+----------------+
|    4820.000000 |
+----------------+
1 row in set (0.01 sec)
```

2. Write a SQL query to Retrieve the top 10 highest account balances



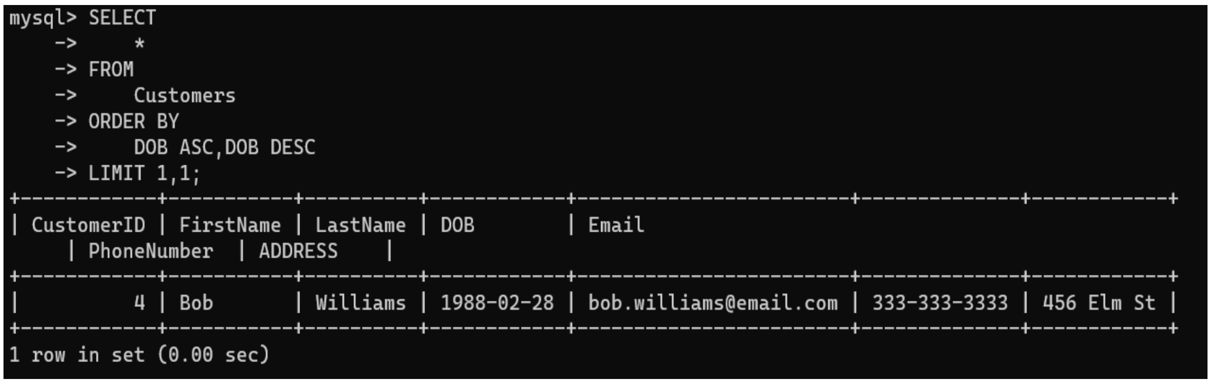
```
mysql> SELECT
    ->     AccountID,
    ->     AccountType,
    ->     Balance
    -> FROM
    ->     Accounts
    -> ORDER BY
    ->     Balance DESC
    -> LIMIT 10;
+-----------+-------------+----------+
| AccountID | AccountType | Balance  |
+-----------+-------------+----------+
|         9 | Savings     | 10000.00 |
|         3 | Savings     |  8000.00 |
|         6 | Checking    |  7000.00 |
|         8 | Checking    |  6000.00 |
|         1 | Savings     |  5000.00 |
|         7 | Savings     |  4000.00 |
|         5 | Savings     |  3000.00 |
|         2 | Checking    |  2500.00 |
|        10 | Checking    |  1500.00 |
|         4 | Checking    |  1200.00 |
+-----------+-------------+----------+
10 rows in set (0.00 sec)
```

3. Write a SQL query to Calculate Total Deposits for All Customers in specific date



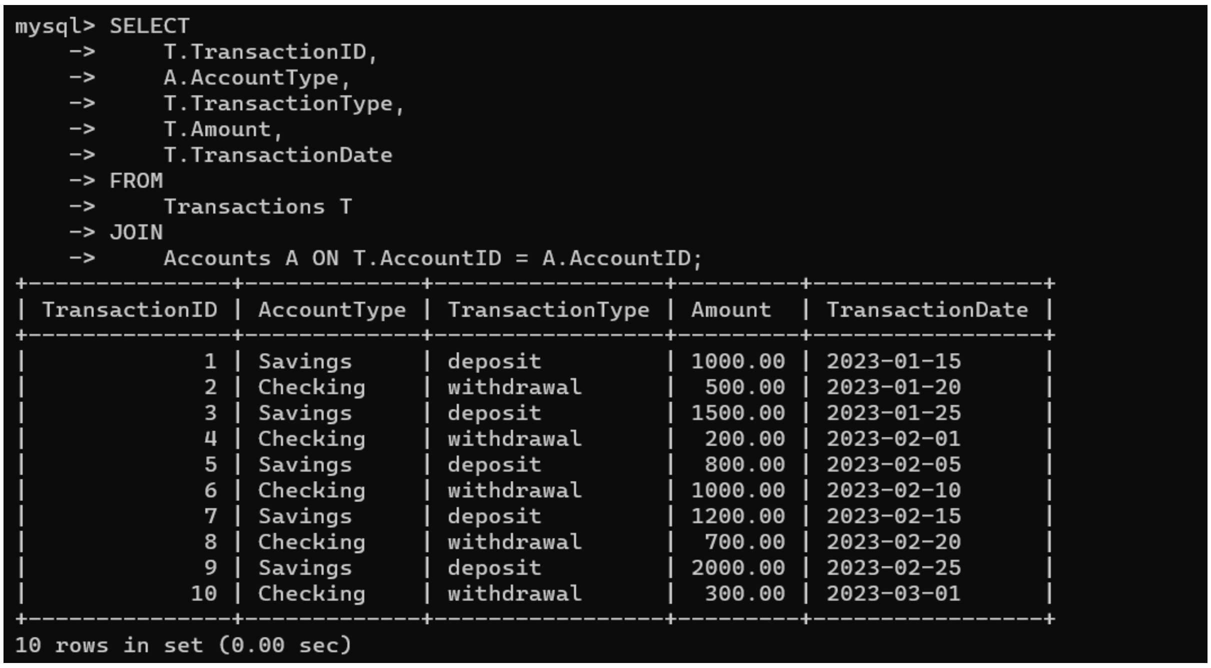
```
mysql> SELECT
    ->     T.TransactionDate,
    ->     SUM(CASE WHEN T.TransactionType = 'Deposit' THEN T.Amount
ELSE 0 END) AS TotalDeposits
    -> FROM
    ->     Transactions T
    -> JOIN
    ->     Accounts A ON T.AccountID = A.AccountID
    -> GROUP BY
    ->     T.TransactionDate;
+-----------------+---------------+
| TransactionDate | TotalDeposits |
+-----------------+---------------+
| 2023-01-15      |       1000.00 |
| 2023-01-20      |          0.00 |
| 2023-01-25      |       1500.00 |
| 2023-02-01      |          0.00 |
| 2023-02-05      |        800.00 |
| 2023-02-10      |          0.00 |
| 2023-02-15      |       1200.00 |
| 2023-02-20      |          0.00 |
| 2023-02-25      |       2000.00 |
| 2023-03-01      |          0.00 |
+-----------------+---------------+
10 rows in set (0.01 sec)
```

4. Write a SQL query to Find the Oldest and Newest Customers.



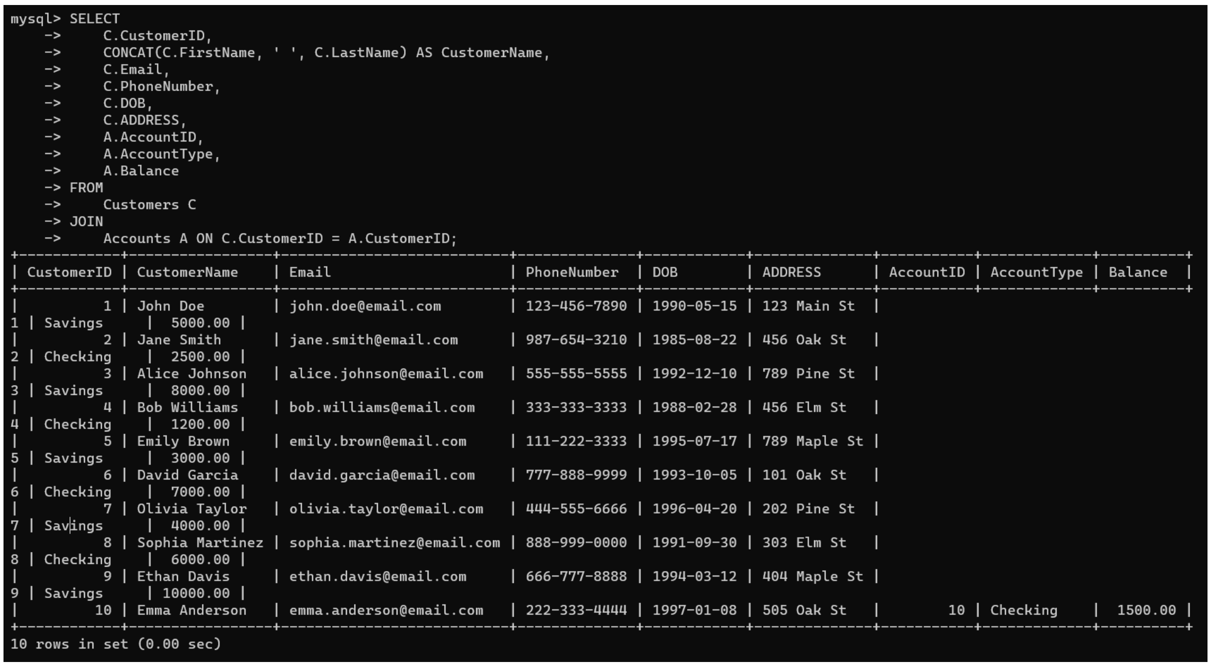
```
mysql> SELECT
    ->     *
    -> FROM
    ->     Customers
    -> ORDER BY
    ->     DOB ASC,DOB DESC
    -> LIMIT 1,1;
+------------+-----------+----------+------------+------------------------+--------------+------------+
| CustomerID | FirstName | LastName | DOB        | Email
     | PhoneNumber  | ADDRESS    |
+------------+-----------+----------+------------+------------------------+--------------+------------+
|          4 | Bob       | Williams | 1988-02-28 | bob.williams@email.com | 333-333-3333 | 456 Elm St |
+------------+-----------+----------+------------+------------------------+--------------+------------+
1 row in set (0.00 sec)
```

5. Write a SQL query to Retrieve transaction details along with the account type.



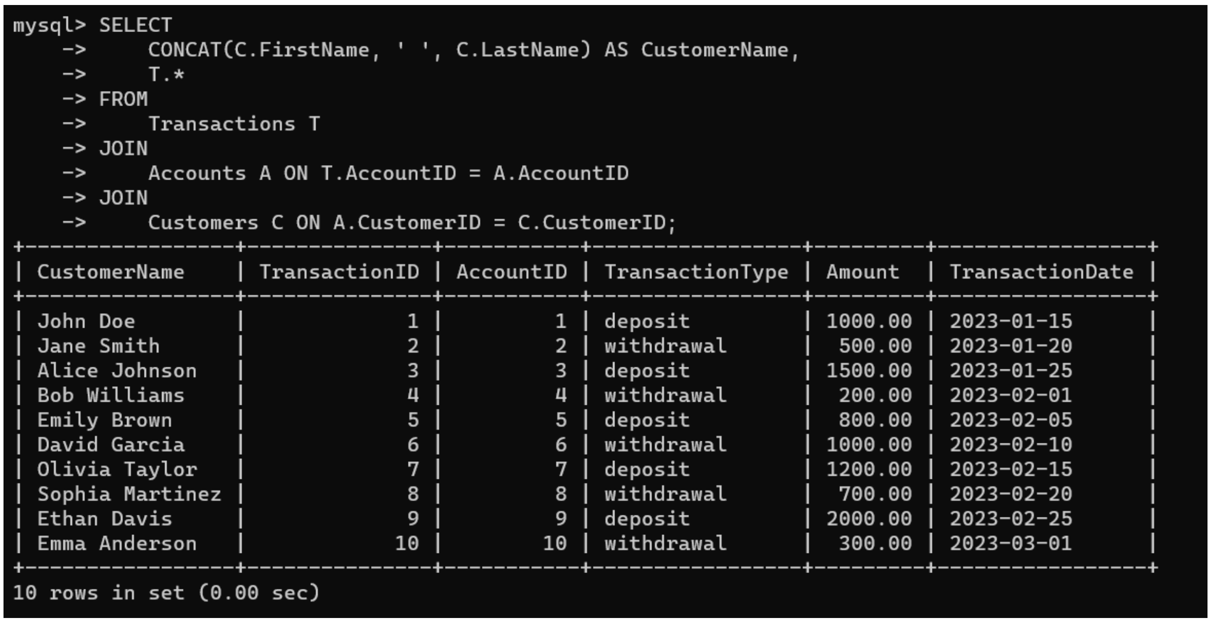
```
mysql> SELECT
    ->     T.TransactionID,
    ->     A.AccountType,
    ->     T.TransactionType,
    ->     T.Amount,
    ->     T.TransactionDate
    -> FROM
    ->     Transactions T
    -> JOIN
    ->     Accounts A ON T.AccountID = A.AccountID;
+---------------+-------------+-----------------+---------+-----------------+
| TransactionID | AccountType | TransactionType | Amount  | TransactionDate |
+---------------+-------------+-----------------+---------+-----------------+
|             1 | Savings     | deposit         | 1000.00 | 2023-01-15      |
|             2 | Checking    | withdrawal      |  500.00 | 2023-01-20      |
|             3 | Savings     | deposit         | 1500.00 | 2023-01-25      |
|             4 | Checking    | withdrawal      |  200.00 | 2023-02-01      |
|             5 | Savings     | deposit         |  800.00 | 2023-02-05      |
|             6 | Checking    | withdrawal      | 1000.00 | 2023-02-10      |
|             7 | Savings     | deposit         | 1200.00 | 2023-02-15      |
|             8 | Checking    | withdrawal      |  700.00 | 2023-02-20      |
|             9 | Savings     | deposit         | 2000.00 | 2023-02-25      |
|            10 | Checking    | withdrawal      |  300.00 | 2023-03-01      |
+---------------+-------------+-----------------+---------+-----------------+
10 rows in set (0.00 sec)
```

6. Write a SQL query to Get a list of customers along with their account details.



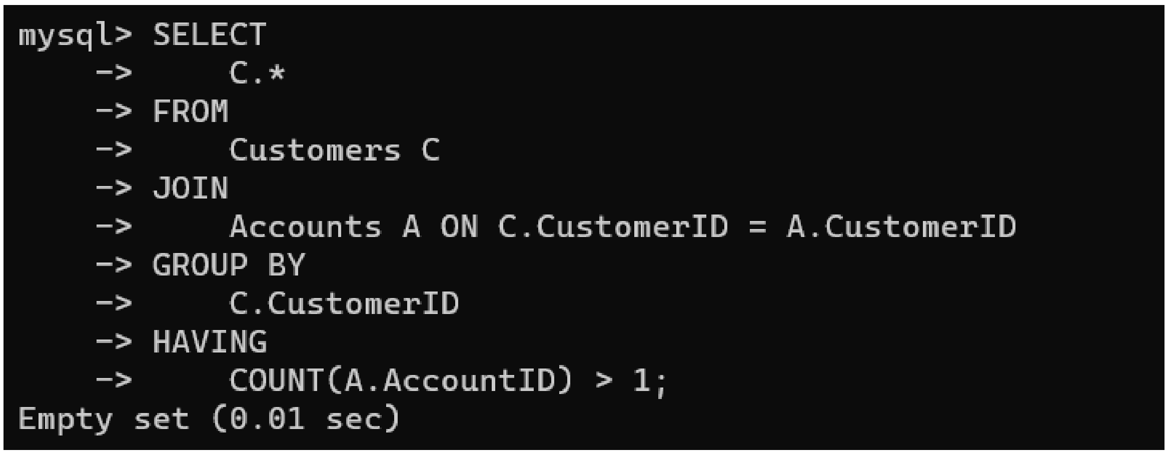
```
mysql> SELECT
    ->     C.CustomerID,
    ->     CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName,
    ->     C.Email,
    ->     C.PhoneNumber,
    ->     C.DOB,
    ->     C.ADDRESS,
    ->     A.AccountID,
    ->     A.AccountType,
    ->     A.Balance
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Accounts A ON C.CustomerID = A.CustomerID;
+------------+-----------------+---------------------------+--------------+------------+--------------+-----------+-------------+----------+
| CustomerID | CustomerName    | Email                     | PhoneNumber  | DOB        | ADDRESS      | AccountID | AccountType | Balance  |
+------------+-----------------+---------------------------+--------------+------------+--------------+-----------+-------------+----------+
|          1 | John Doe        | john.doe@email.com        | 123-456-7890 | 1990-05-15 | 123 Main St  |
1 | Savings     |  5000.00 |
|          2 | Jane Smith      | jane.smith@email.com      | 987-654-3210 | 1985-08-22 | 456 Oak St   |
2 | Checking    |  2500.00 |
|          3 | Alice Johnson   | alice.johnson@email.com   | 555-555-5555 | 1992-12-10 | 789 Pine St  |
3 | Savings     |  8000.00 |
|          4 | Bob Williams    | bob.williams@email.com    | 333-333-3333 | 1988-02-28 | 456 Elm St   |
4 | Checking    |  1200.00 |
|          5 | Emily Brown     | emily.brown@email.com     | 111-222-3333 | 1995-07-17 | 789 Maple St |
5 | Savings     |  3000.00 |
|          6 | David Garcia    | david.garcia@email.com    | 777-888-9999 | 1993-10-05 | 101 Oak St   |
6 | Checking    |  7000.00 |
|          7 | Olivia Taylor   | olivia.taylor@email.com   | 444-555-6666 | 1996-04-20 | 202 Pine St  |
7 | Savings     |  4000.00 |
|          8 | Sophia Martinez | sophia.martinez@email.com | 888-999-0000 | 1991-09-30 | 303 Elm St   |
8 | Checking    |  6000.00 |
|          9 | Ethan Davis     | ethan.davis@email.com     | 666-777-8888 | 1994-03-12 | 404 Maple St |
9 | Savings     | 10000.00 |
|         10 | Emma Anderson   | emma.anderson@email.com   | 222-333-4444 | 1997-01-08 | 505 Oak St   |        10 | Checking    |  1500.00 |
+------------+-----------------+---------------------------+--------------+------------+--------------+-----------+-------------+----------+
10 rows in set (0.00 sec)
```

7. Write a SQL query to Retrieve transaction details along with customer information for a specific account.



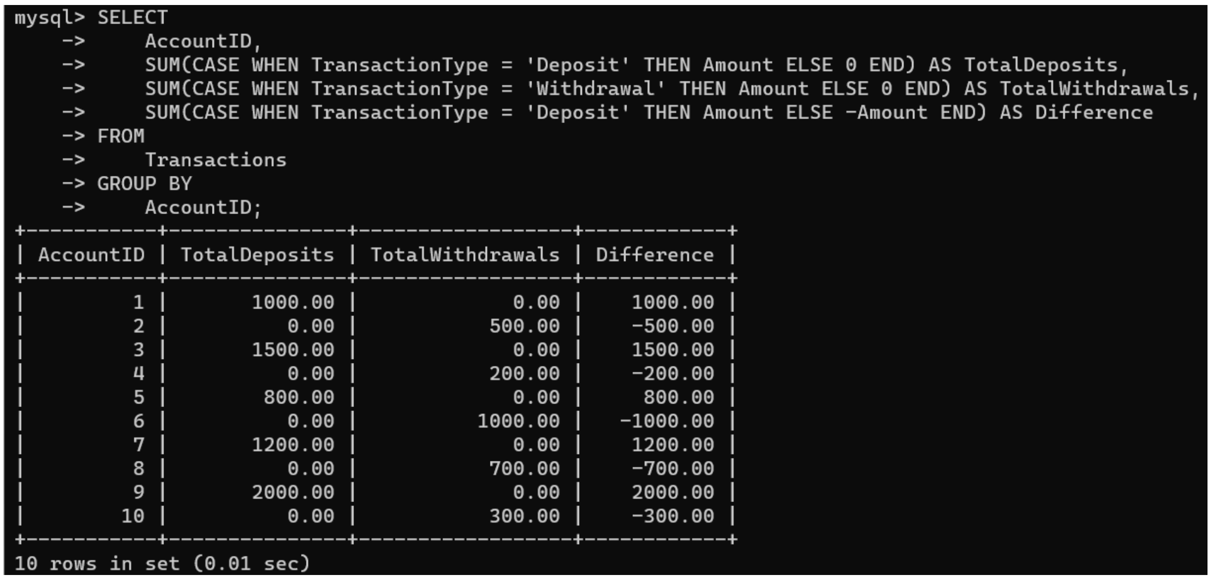
```
mysql> SELECT
    ->     CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName,
    ->     T.*
    -> FROM
    ->     Transactions T
    -> JOIN
    ->     Accounts A ON T.AccountID = A.AccountID
    -> JOIN
    ->     Customers C ON A.CustomerID = C.CustomerID;
+-----------------+---------------+-----------+-----------------+---------+-----------------+
| CustomerName    | TransactionID | AccountID | TransactionType | Amount  | TransactionDate |
+-----------------+---------------+-----------+-----------------+---------+-----------------+
| John Doe        |             1 |         1 | deposit         | 1000.00 | 2023-01-15      |
| Jane Smith      |             2 |         2 | withdrawal      |  500.00 | 2023-01-20      |
| Alice Johnson   |             3 |         3 | deposit         | 1500.00 | 2023-01-25      |
| Bob Williams    |             4 |         4 | withdrawal      |  200.00 | 2023-02-01      |
| Emily Brown     |             5 |         5 | deposit         |  800.00 | 2023-02-05      |
| David Garcia    |             6 |         6 | withdrawal      | 1000.00 | 2023-02-10      |
| Olivia Taylor   |             7 |         7 | deposit         | 1200.00 | 2023-02-15      |
| Sophia Martinez |             8 |         8 | withdrawal      |  700.00 | 2023-02-20      |
| Ethan Davis     |             9 |         9 | deposit         | 2000.00 | 2023-02-25      |
| Emma Anderson   |            10 |        10 | withdrawal      |  300.00 | 2023-03-01      |
+-----------------+---------------+-----------+-----------------+---------+-----------------+
10 rows in set (0.00 sec)
```

8. Write a SQL query to Identify customers who have more than one account.



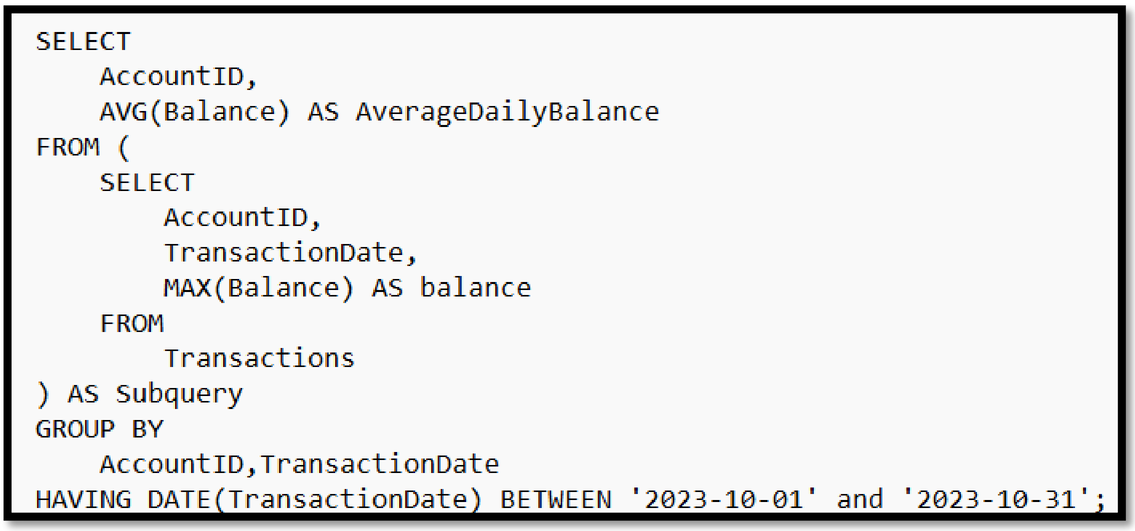
```
mysql> SELECT
    ->     C.*
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Accounts A ON C.CustomerID = A.CustomerID
    -> GROUP BY
    ->     C.CustomerID
    -> HAVING
    ->     COUNT(A.AccountID) > 1;
Empty set (0.01 sec)
```

9. Write a SQL query to Calculate the difference in transaction amounts between deposits and withdrawals.



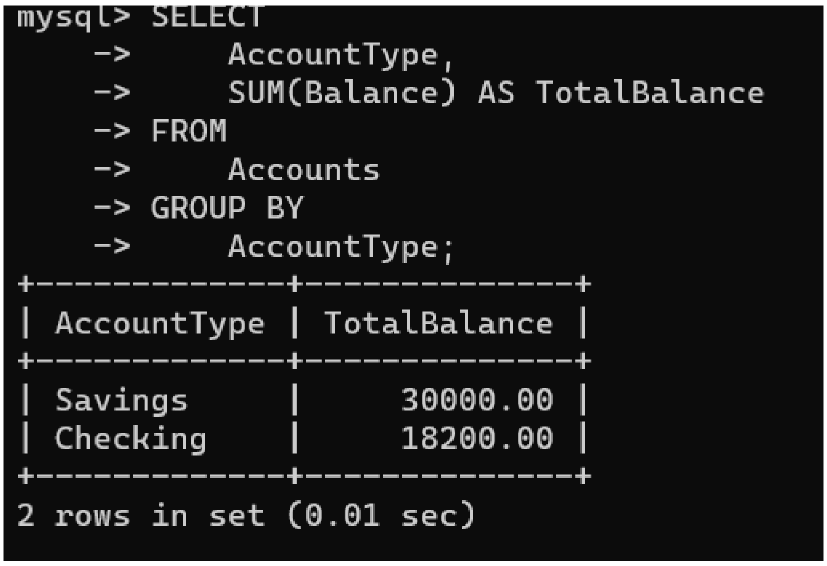
```
mysql> SELECT
    ->     AccountID,
    ->     SUM(CASE WHEN TransactionType = 'Deposit' THEN Amount ELSE 0 END) AS TotalDeposits,
    ->     SUM(CASE WHEN TransactionType = 'Withdrawal' THEN Amount ELSE 0 END) AS TotalWithdrawals,
    ->     SUM(CASE WHEN TransactionType = 'Deposit' THEN Amount ELSE -Amount END) AS Difference
    -> FROM
    ->     Transactions
    -> GROUP BY
    ->     AccountID;
+-----------+---------------+------------------+------------+
| AccountID | TotalDeposits | TotalWithdrawals | Difference |
+-----------+---------------+------------------+------------+
|         1 |       1000.00 |             0.00 |    1000.00 |
|         2 |          0.00 |           500.00 |    -500.00 |
|         3 |       1500.00 |             0.00 |    1500.00 |
|         4 |          0.00 |           200.00 |    -200.00 |
|         5 |        800.00 |             0.00 |     800.00 |
|         6 |          0.00 |          1000.00 |   -1000.00 |
|         7 |       1200.00 |             0.00 |    1200.00 |
|         8 |          0.00 |           700.00 |    -700.00 |
|         9 |       2000.00 |             0.00 |    2000.00 |
|        10 |          0.00 |           300.00 |    -300.00 |
+-----------+---------------+------------------+------------+
10 rows in set (0.01 sec)
```

10. Write a SQL query to Calculate the average daily balance for each account over a specified period.



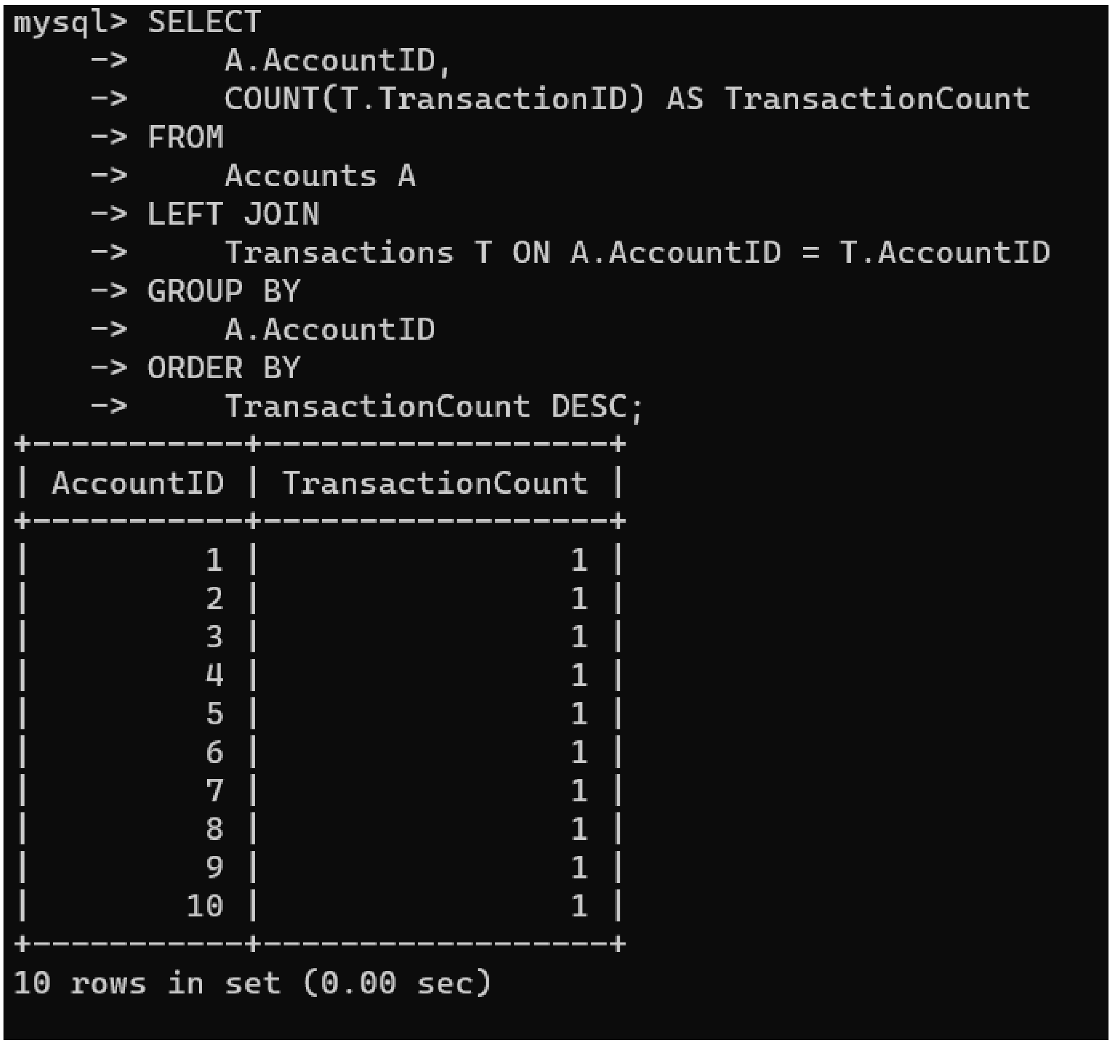
```
SELECT
    AccountID,
    AVG(Balance) AS AverageDailyBalance
FROM (
    SELECT
        AccountID,
        TransactionDate,
        MAX(Balance) AS balance
    FROM
        Transactions
) AS Subquery
GROUP BY
    AccountID,TransactionDate
HAVING DATE(TransactionDate) BETWEEN '2023-10-01' and '2023-10-31';
```

11. Calculate the total balance for each account type.



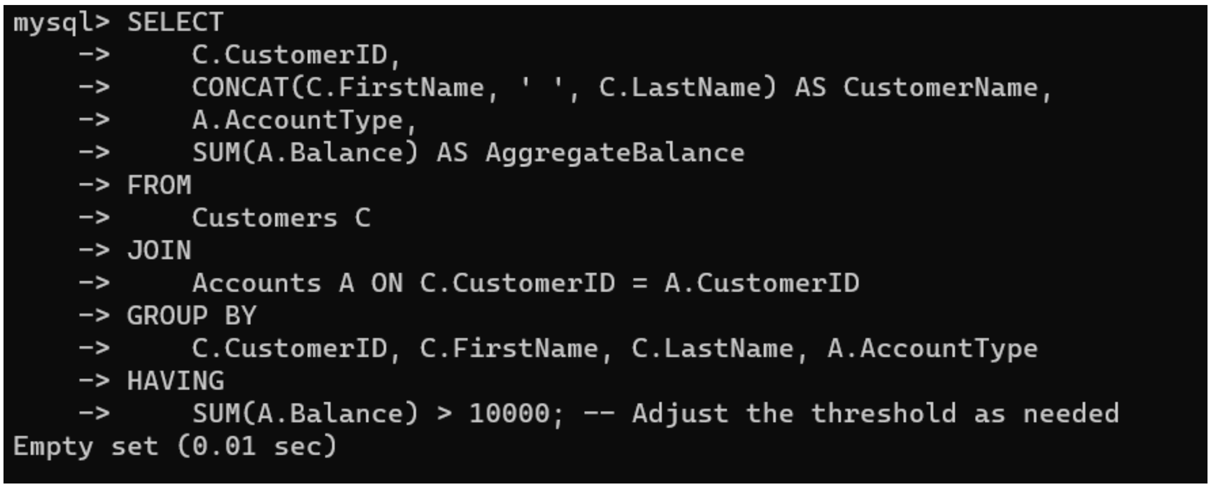
```
mysql> SELECT
    ->     AccountType,
    ->     SUM(Balance) AS TotalBalance
    -> FROM
    ->     Accounts
    -> GROUP BY
    ->     AccountType;
+-------------+--------------+
| AccountType | TotalBalance |
+-------------+--------------+
| Savings     |     30000.00 |
| Checking    |     18200.00 |
+-------------+--------------+
2 rows in set (0.01 sec)
```

12. Identify accounts with the highest number of transactions order by descending order.



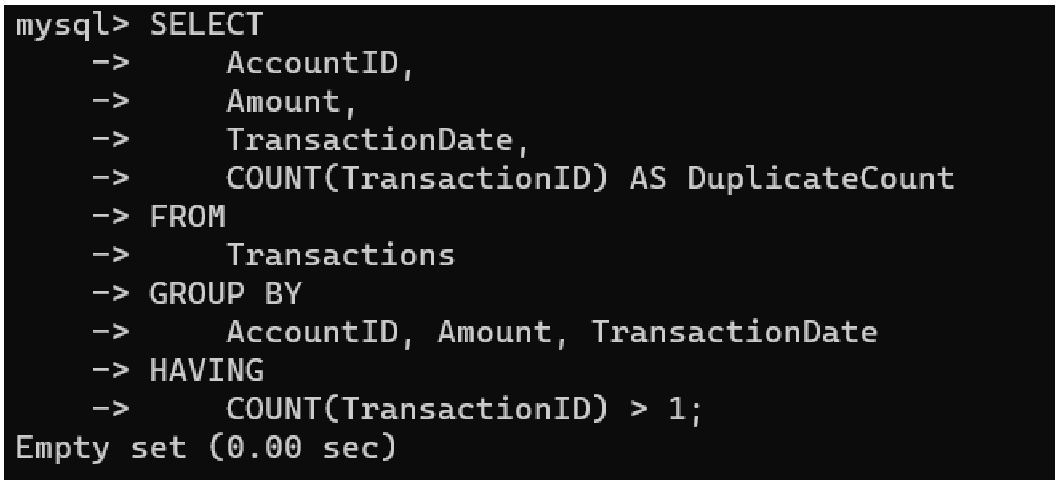
```
mysql> SELECT
    ->     A.AccountID,
    ->     COUNT(T.TransactionID) AS TransactionCount
    -> FROM
    ->     Accounts A
    -> LEFT JOIN
    ->     Transactions T ON A.AccountID = T.AccountID
    -> GROUP BY
    ->     A.AccountID
    -> ORDER BY
    ->     TransactionCount DESC;
+-----------+------------------+
| AccountID | TransactionCount |
+-----------+------------------+
|         1 |                1 |
|         2 |                1 |
|         3 |                1 |
|         4 |                1 |
|         5 |                1 |
|         6 |                1 |
|         7 |                1 |
|         8 |                1 |
|         9 |                1 |
|        10 |                1 |
+-----------+------------------+
10 rows in set (0.00 sec)
```

13. List customers with high aggregate account balances, along with their account types.



```
mysql> SELECT
    ->     C.CustomerID,
    ->     CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName,
    ->     A.AccountType,
    ->     SUM(A.Balance) AS AggregateBalance
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Accounts A ON C.CustomerID = A.CustomerID
    -> GROUP BY
    ->     C.CustomerID, C.FirstName, C.LastName, A.AccountType
    -> HAVING
    ->     SUM(A.Balance) > 10000; -- Adjust the threshold as needed
Empty set (0.01 sec)
```

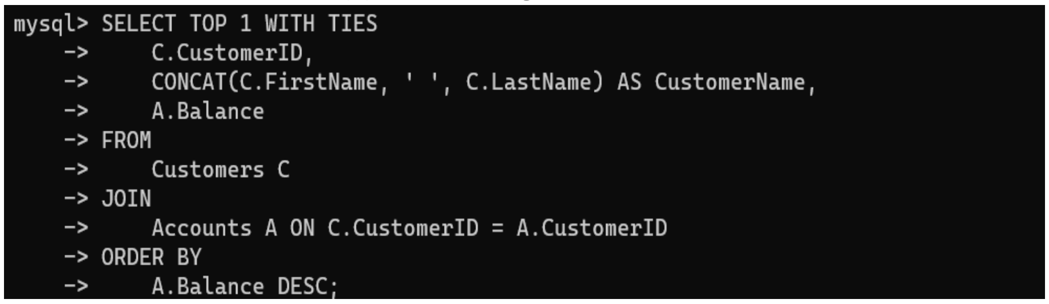
14. Identify and list duplicate transactions based on transaction amount, date, and account.



```
mysql> SELECT
    ->     AccountID,
    ->     Amount,
    ->     TransactionDate,
    ->     COUNT(TransactionID) AS DuplicateCount
    -> FROM
    ->     Transactions
    -> GROUP BY
    ->     AccountID, Amount, TransactionDate
    -> HAVING
    ->     COUNT(TransactionID) > 1;
Empty set (0.00 sec)
```

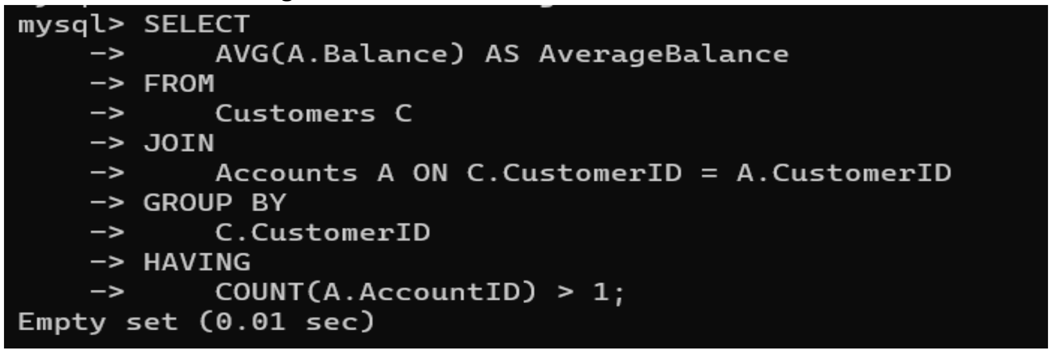
Tasks 4: Subquery and its type:

1. Retrieve the customer(s) with the highest account balance.



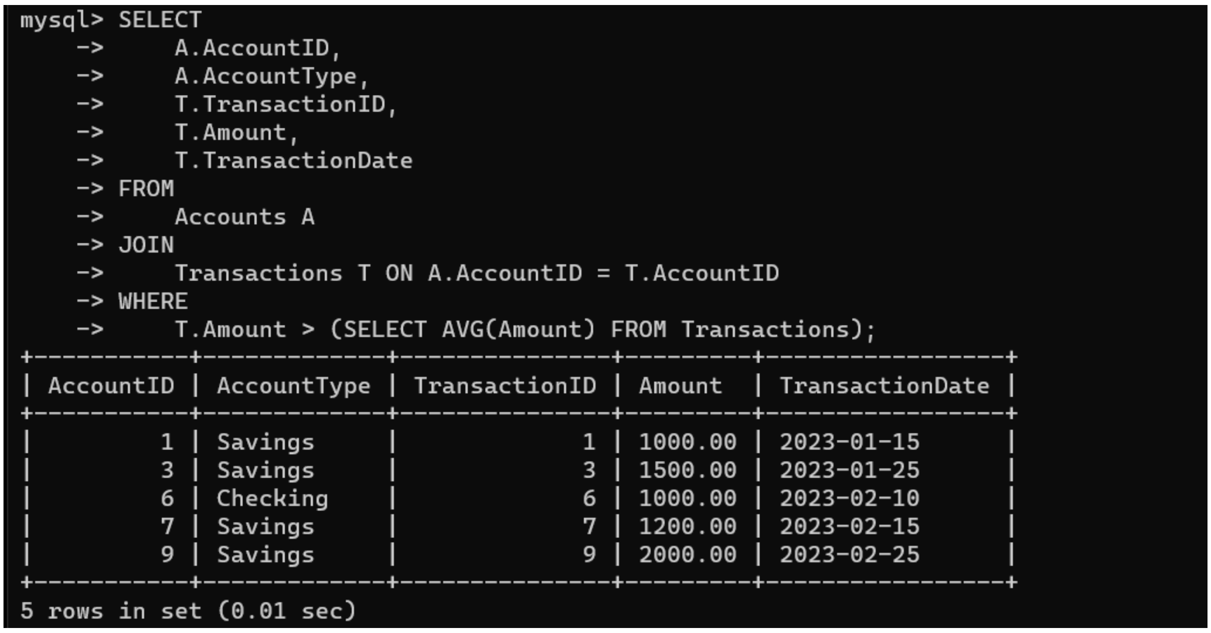
```
mysql> SELECT TOP 1 WITH TIES
    ->     C.CustomerID,
    ->     CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName,
    ->     A.Balance
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Accounts A ON C.CustomerID = A.CustomerID
    -> ORDER BY
    ->     A.Balance DESC;
```

2. Calculate the average account balance for customers who have more than one account.



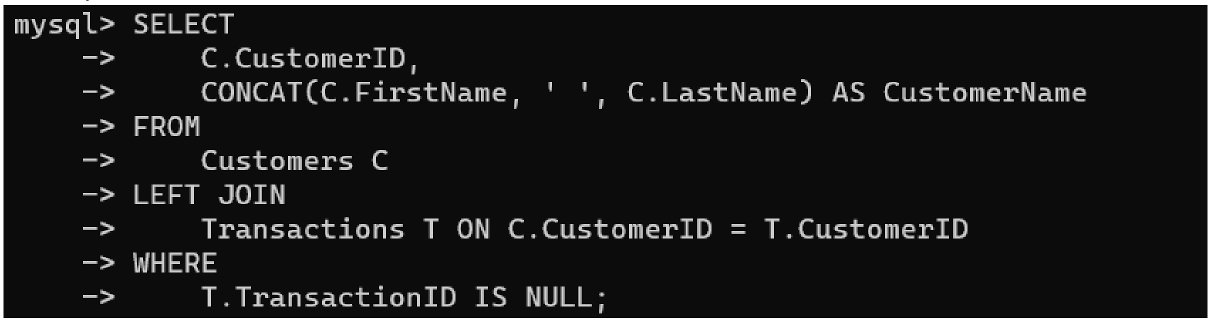
```
mysql> SELECT
    ->     AVG(A.Balance) AS AverageBalance
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Accounts A ON C.CustomerID = A.CustomerID
    -> GROUP BY
    ->     C.CustomerID
    -> HAVING
    ->     COUNT(A.AccountID) > 1;
Empty set (0.01 sec)
```

3. Retrieve accounts with transactions whose amounts exceed the average transaction amount.



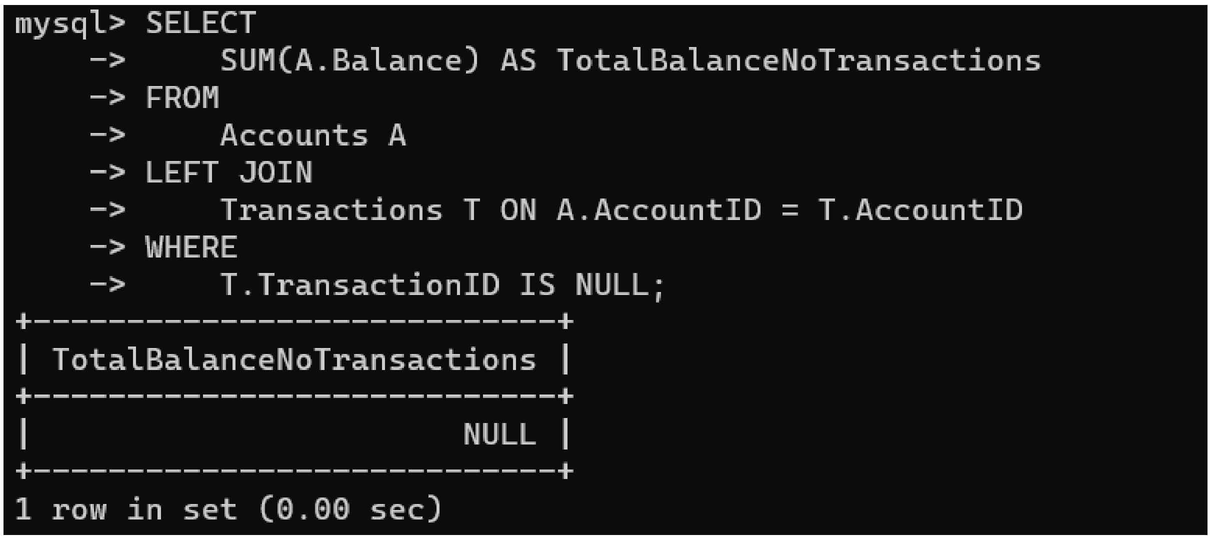
```
mysql> SELECT
    ->     A.AccountID,
    ->     A.AccountType,
    ->     T.TransactionID,
    ->     T.Amount,
    ->     T.TransactionDate
    -> FROM
    ->     Accounts A
    -> JOIN
    ->     Transactions T ON A.AccountID = T.AccountID
    -> WHERE
    ->     T.Amount > (SELECT AVG(Amount) FROM Transactions);
+-----------+-------------+---------------+---------+-----------------+
| AccountID | AccountType | TransactionID | Amount  | TransactionDate |
+-----------+-------------+---------------+---------+-----------------+
|         1 | Savings     |             1 | 1000.00 | 2023-01-15      |
|         3 | Savings     |             3 | 1500.00 | 2023-01-25      |
|         6 | Checking    |             6 | 1000.00 | 2023-02-10      |
|         7 | Savings     |             7 | 1200.00 | 2023-02-15      |
|         9 | Savings     |             9 | 2000.00 | 2023-02-25      |
+-----------+-------------+---------------+---------+-----------------+
5 rows in set (0.01 sec)
```

4. Identify customers who have no recorded transactions.



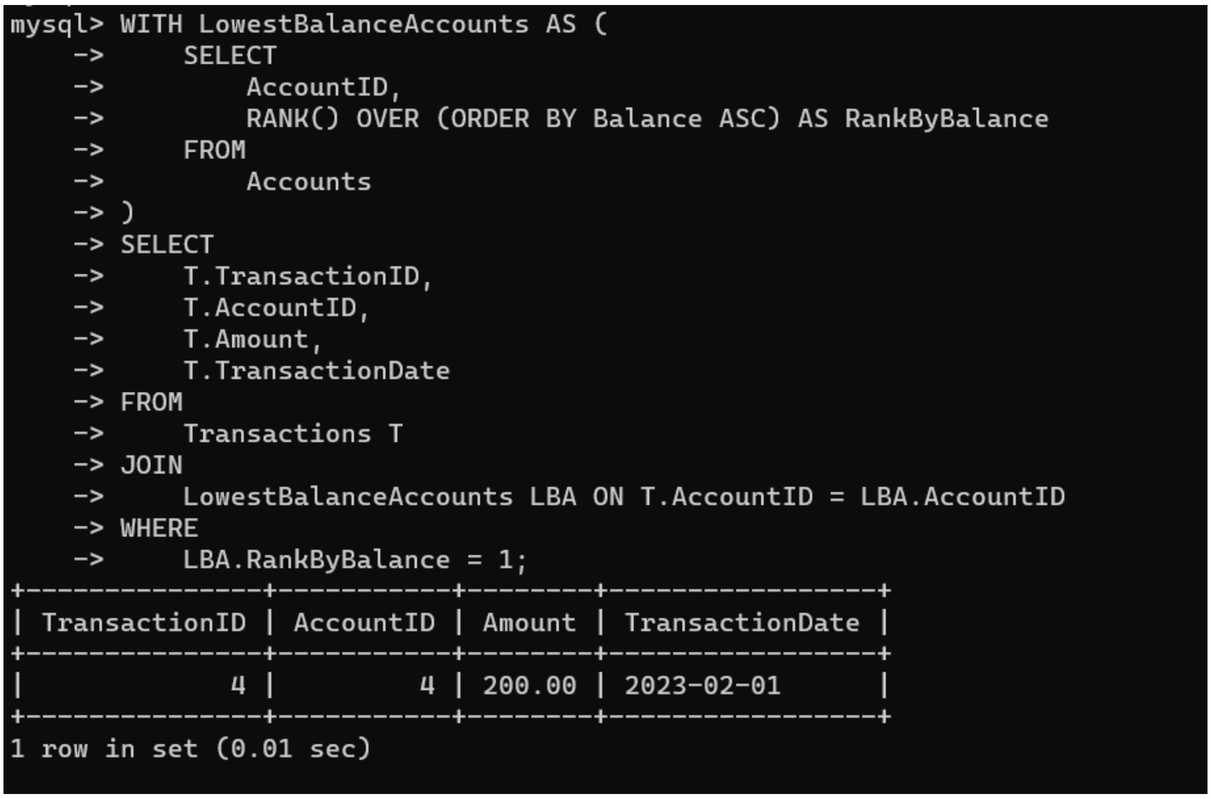
```
mysql> SELECT
    ->     C.CustomerID,
    ->     CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName
    -> FROM
    ->     Customers C
    -> LEFT JOIN
    ->     Transactions T ON C.CustomerID = T.CustomerID
    -> WHERE
    ->     T.TransactionID IS NULL;
```

5. Calculate the total balance of accounts with no recorded transactions.



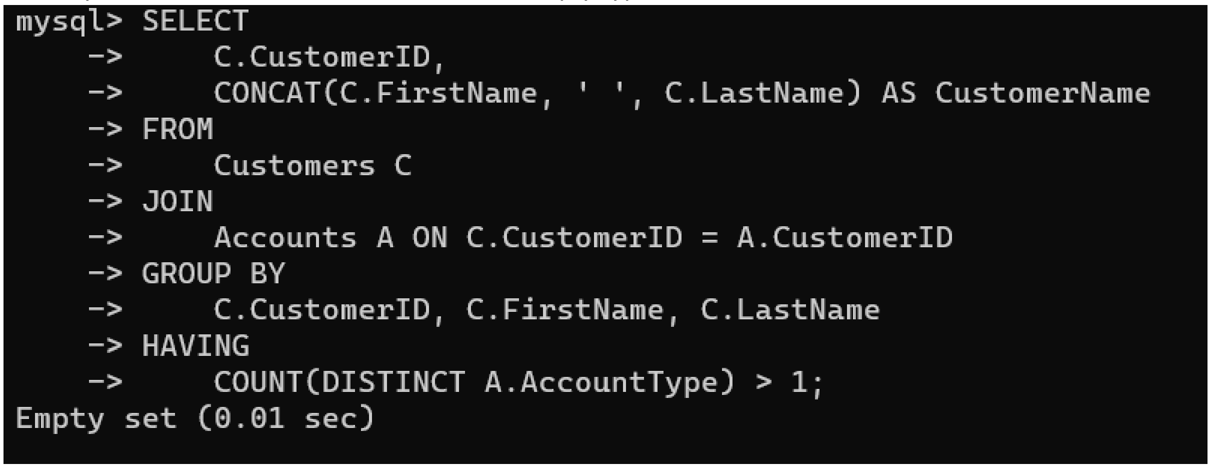
```
mysql> SELECT
    ->     SUM(A.Balance) AS TotalBalanceNoTransactions
    -> FROM
    ->     Accounts A
    -> LEFT JOIN
    ->     Transactions T ON A.AccountID = T.AccountID
    -> WHERE
    ->     T.TransactionID IS NULL;
+----------------------------+
| TotalBalanceNoTransactions |
+----------------------------+
|                       NULL |
+----------------------------+
1 row in set (0.00 sec)
```

6. Retrieve transactions for accounts with the lowest balance.



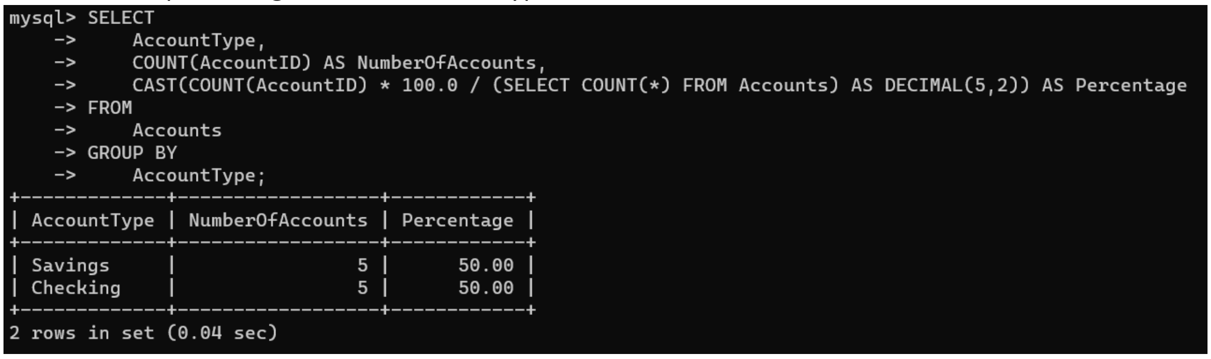
```
mysql> WITH LowestBalanceAccounts AS (
    ->     SELECT
    ->         AccountID,
    ->         RANK() OVER (ORDER BY Balance ASC) AS RankByBalance
    ->     FROM
    ->         Accounts
    -> )
    -> SELECT
    ->     T.TransactionID,
    ->     T.AccountID,
    ->     T.Amount,
    ->     T.TransactionDate
    -> FROM
    ->     Transactions T
    -> JOIN
    ->     LowestBalanceAccounts LBA ON T.AccountID = LBA.AccountID
    -> WHERE
    ->     LBA.RankByBalance = 1;
+---------------+-----------+--------+-----------------+
| TransactionID | AccountID | Amount | TransactionDate |
+---------------+-----------+--------+-----------------+
|             4 |         4 | 200.00 | 2023-02-01      |
+---------------+-----------+--------+-----------------+
1 row in set (0.01 sec)
```

7. Identify customers who have accounts of multiply types.



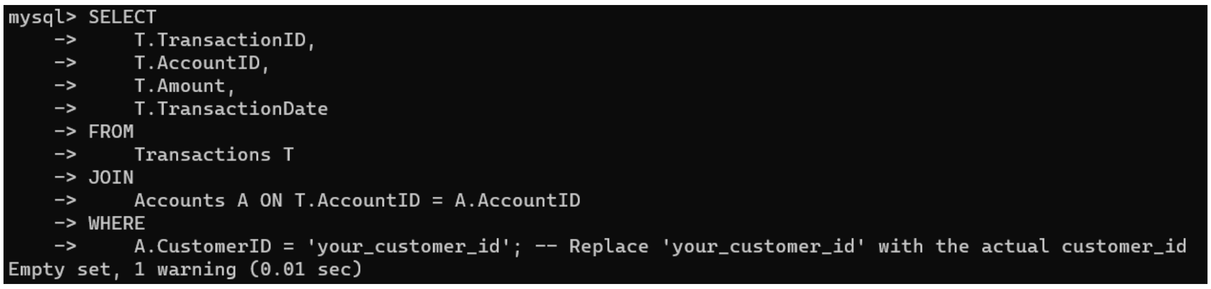
```
mysql> SELECT
    ->     C.CustomerID,
    ->     CONCAT(C.FirstName, ' ', C.LastName) AS CustomerName
    -> FROM
    ->     Customers C
    -> JOIN
    ->     Accounts A ON C.CustomerID = A.CustomerID
    -> GROUP BY
    ->     C.CustomerID, C.FirstName, C.LastName
    -> HAVING
    ->     COUNT(DISTINCT A.AccountType) > 1;
Empty set (0.01 sec)
```

8. Calculate the percentage of each account type out of the total number of accounts.



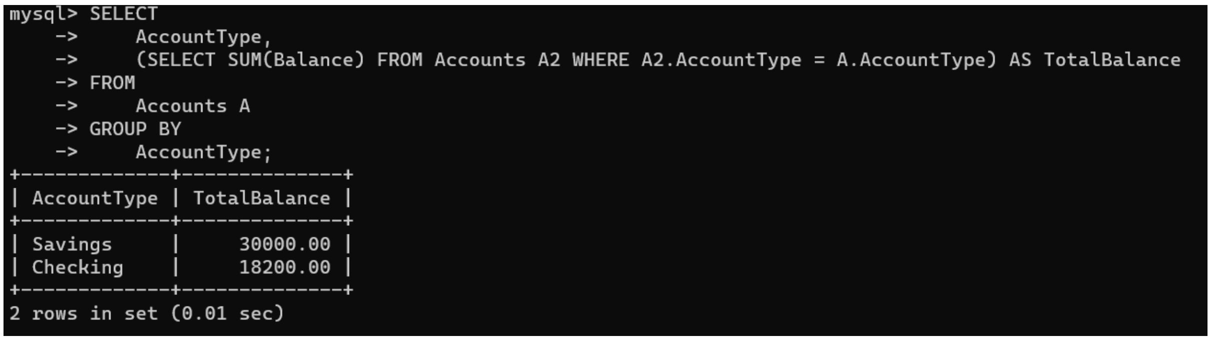
```
mysql> SELECT
    ->     AccountType,
    ->     COUNT(AccountID) AS NumberOfAccounts,
    ->     CAST(COUNT(AccountID) * 100.0 / (SELECT COUNT(*) FROM Accounts) AS DECIMAL(5,2)) AS Percentage
    -> FROM
    ->     Accounts
    -> GROUP BY
    ->     AccountType;
+-------------+------------------+------------+
| AccountType | NumberOfAccounts | Percentage |
+-------------+------------------+------------+
| Savings     |                5 |      50.00 |
| Checking    |                5 |      50.00 |
+-------------+------------------+------------+
2 rows in set (0.04 sec)
```

9. Retrieve all transactions for a customer with a given customer_id.



```
mysql> SELECT
    ->     T.TransactionID,
    ->     T.AccountID,
    ->     T.Amount,
    ->     T.TransactionDate
    -> FROM
    ->     Transactions T
    -> JOIN
    ->     Accounts A ON T.AccountID = A.AccountID
    -> WHERE
    ->     A.CustomerID = 'your_customer_id'; -- Replace 'your_customer_id' with the actual customer_id
Empty set, 1 warning (0.01 sec)
```

10. Calculate the total balance for each account type, including a subquery within the SELECT clause.



```
mysql> SELECT
    ->     AccountType,
    ->     (SELECT SUM(Balance) FROM Accounts A2 WHERE A2.AccountType = A.AccountType) AS TotalBalance
    -> FROM
    ->     Accounts A
    -> GROUP BY
    ->     AccountType;
+-------------+--------------+
| AccountType | TotalBalance |
+-------------+--------------+
| Savings     |     30000.00 |
| Checking    |     18200.00 |
+-------------+--------------+
2 rows in set (0.01 sec)
```